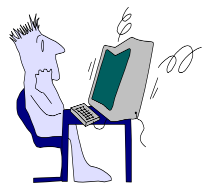


Real Time Safety

Assessment of Software Development Tools for Safety Critical Real-Time Systems

FAA Contract DTFA0301C00048

Project Personnel:

Faculty:	A.J.Kornecki J.Zalewski, N.Brixius
Students:	J.P.Linardon, J.Labbe, L.Crawford, H.Lau, K.Hall, D.Hearn, C.Sanoulliet, M.Milesi

Presented by: Andrew J. Kornecki

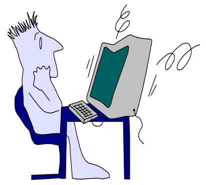
Department of Computer and Software Engineering

Embry Riddle Aeronautical University

Daytona Beach, FL 32114

phone: (386) 226-6888 ; kornecka@erau.edu ; http://faculty.erau.edu/korn

FY2005 Software/Complex Electronic Hardware Standardization Conference, Norfolk, VA, July 26-28, 2005



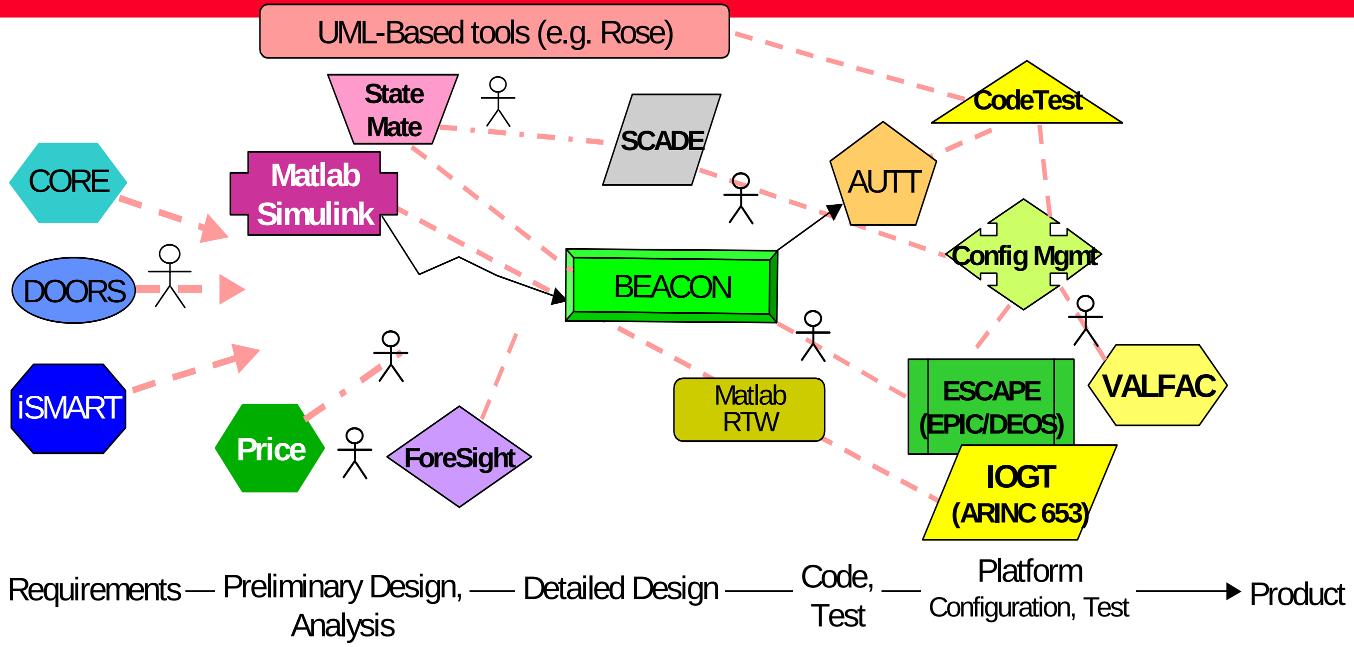
Outline

Real Time Safety

- **Project Background**
- Research Approach
- Development Tool Assessment
- Development Tool Qualification
- Experiments
- Software Tool Forum
- Research Findings
- Conclusions

Real Time Safety

Current Tools: Issues, and Limitations



- Each COTS tool captures a single narrow aspect of the system design/architecture
  - ⇒ System designer must depend on other tools for architectural representation and integration
    Custom workarounds, patches, and conversions needed around each COTS tool, in each project
- System is captured in different forms in different COTS tools
  - ⇒ Manual translation between tools causes lot of duplicated effort
- Proliferation of tools and design notations
  - ⇒ Lack of standard design/architecture notations leads to less re-use across SBUs
    Platform dependencies get embedded in design, reducing portability

SSL SOFTWARE SOLUTIONS LAB

Honeywell

With permission of Honeywell Software Solution Lab

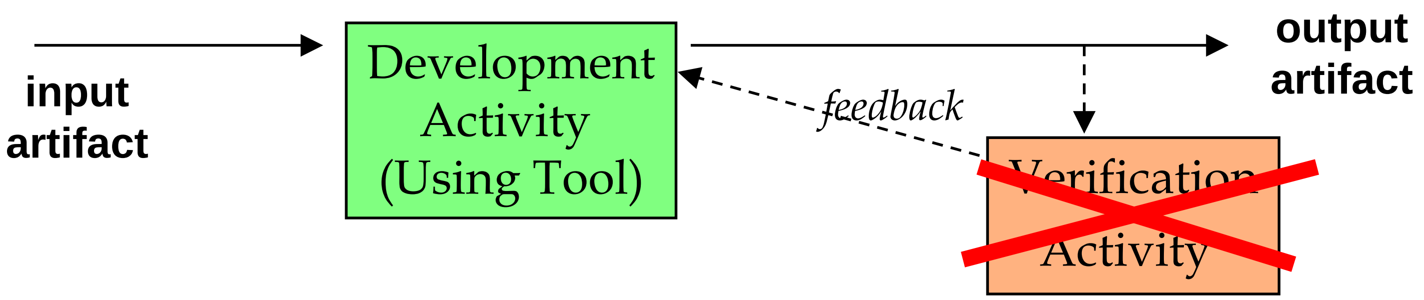
Real Time Safety

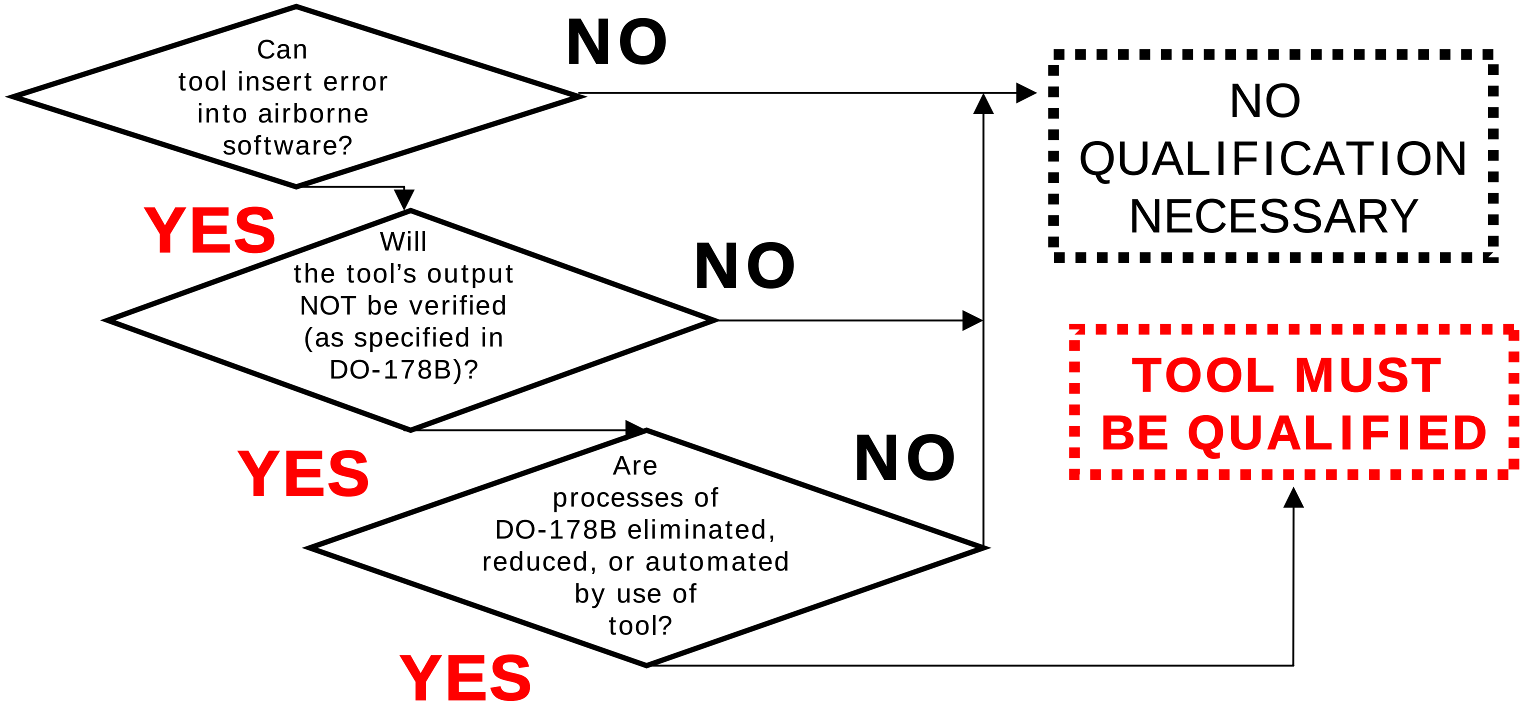
Background: DO-178B Definitions

- ***Software tool**: A computer program used to help develop, test, analyse, produce or modify another program or its documentation*
- ***Software development tools**: Tools whose output is part of airborne software and thus can introduce errors*
- *References:*
  - Section 12.2 of DO-178B
  - Chapter 9 in Order N8110.49 *Software Approval Guidelines*
- NOTE: DO-178B is a “guidance” document only focusing on software processes and objectives to comply with these processes
- The development tools provide a **transformation** between the **input** and **output** artifacts

Real Time Safety

Background: Not Qualified vs. Qualified Development Tools *Section 9.3 of Order N8110.49*




Can tool insert error into airborne software?
NO
YES
Will the tool's output NOT be verified (as specified in DO-178B)?
NO
YES
Are processes of DO-178B eliminated, reduced, or automated by use of tool?
NO
YES
NO QUALIFICATION NECESSARY
TOOL MUST BE QUALIFIED

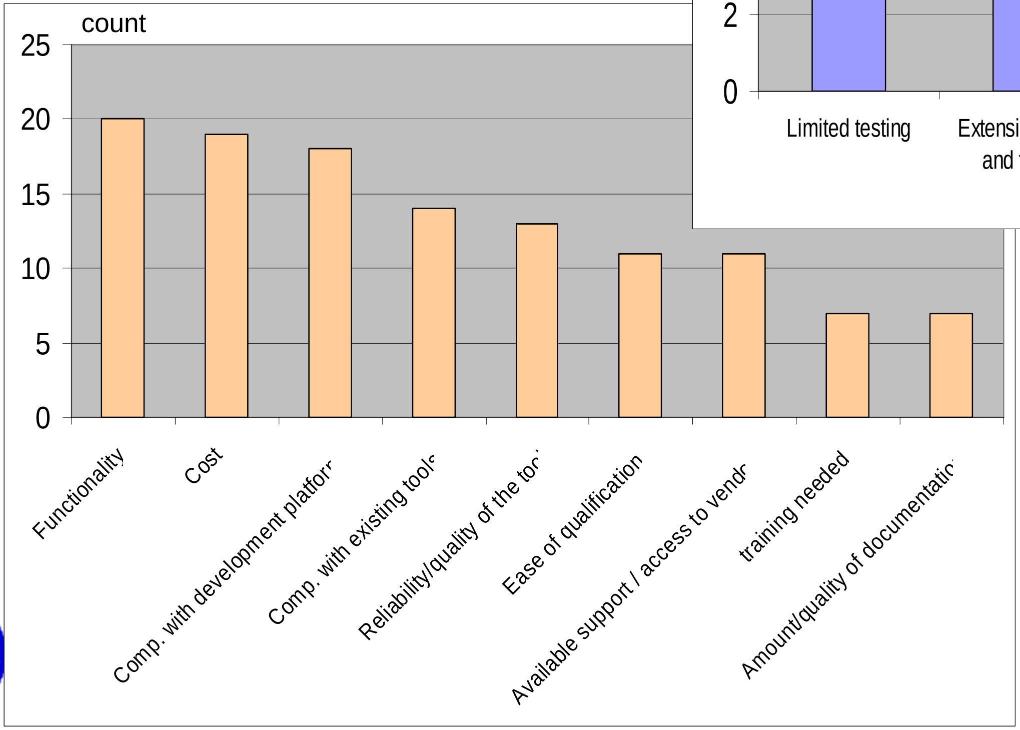
Background: Hypotheses

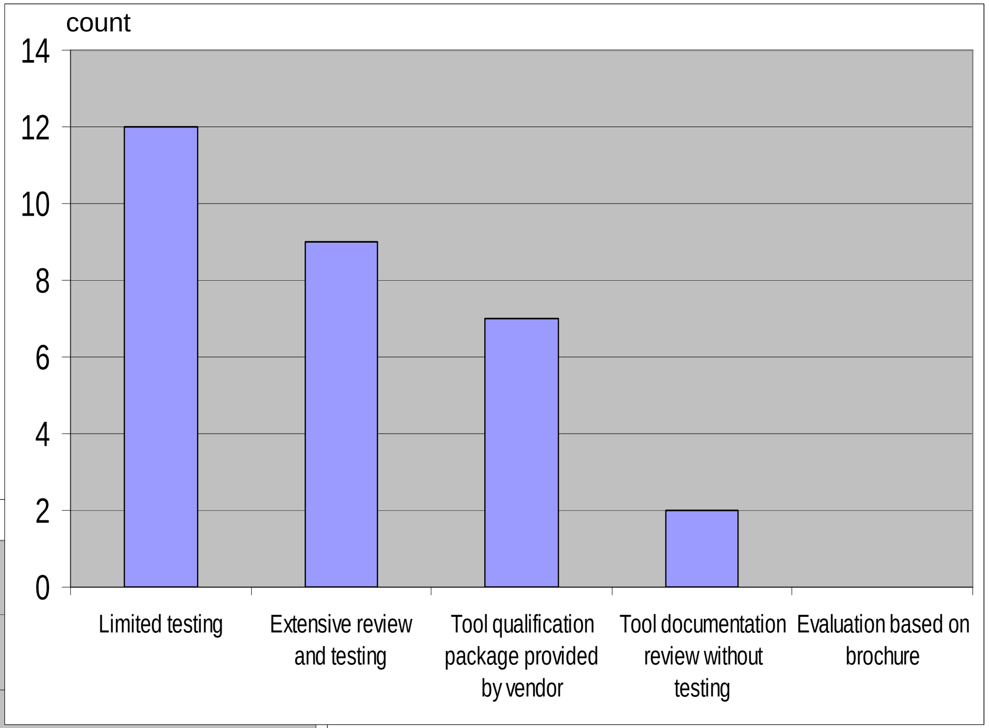
- Theory:
  - o Tools improve development process by automation and reduction of repetitive tasks
  - o Qualified development tools could help with the current certification process by reducing the verification burden of the intermediate software lifecycle artifacts
- Practice:
  - o Although the tool qualification is a well established concept, only a handful of development tools were even attempted to be qualified
- The reasons for this situation have been explored in the research

Real Time Safety

Background: Survey - Results

- 28 respondents representing airborne software (74%) and the FAA personnel (14%) at the FAA Software Conference, May 2002, Dallas/Ft.Worth,





- Follow-up on the e-mail to the FAA Software Certification mailing list (only 14 responses from over 500 individuals)

Real Time Safety

Background: Survey - Major Issues

- Differing perspectives: applicant, certifying authority, tool vendor
- Cost
- Obsolescence
- Tool functionality and compatibility with existing development environment
- Tool vendors typically not used to the level of effort required for a DO-178B compliance
- False vendor claims (scaling, independence)
- Difficult to identify features of a tool which could cause/contribute to errors in the target software
- Inadequate documentation, training and understanding of development tool
- Discouraging rigor of tool qualification and perception of qualification high cost
- Business model prevents from investing in tool qualification
- Need for re-qualification for each new certification project
- Tool reliability as a measure is questionable (How to show it? Does it matter?)
- Vendor support and alternate means for COTS tools

Real Time Safety

Outline

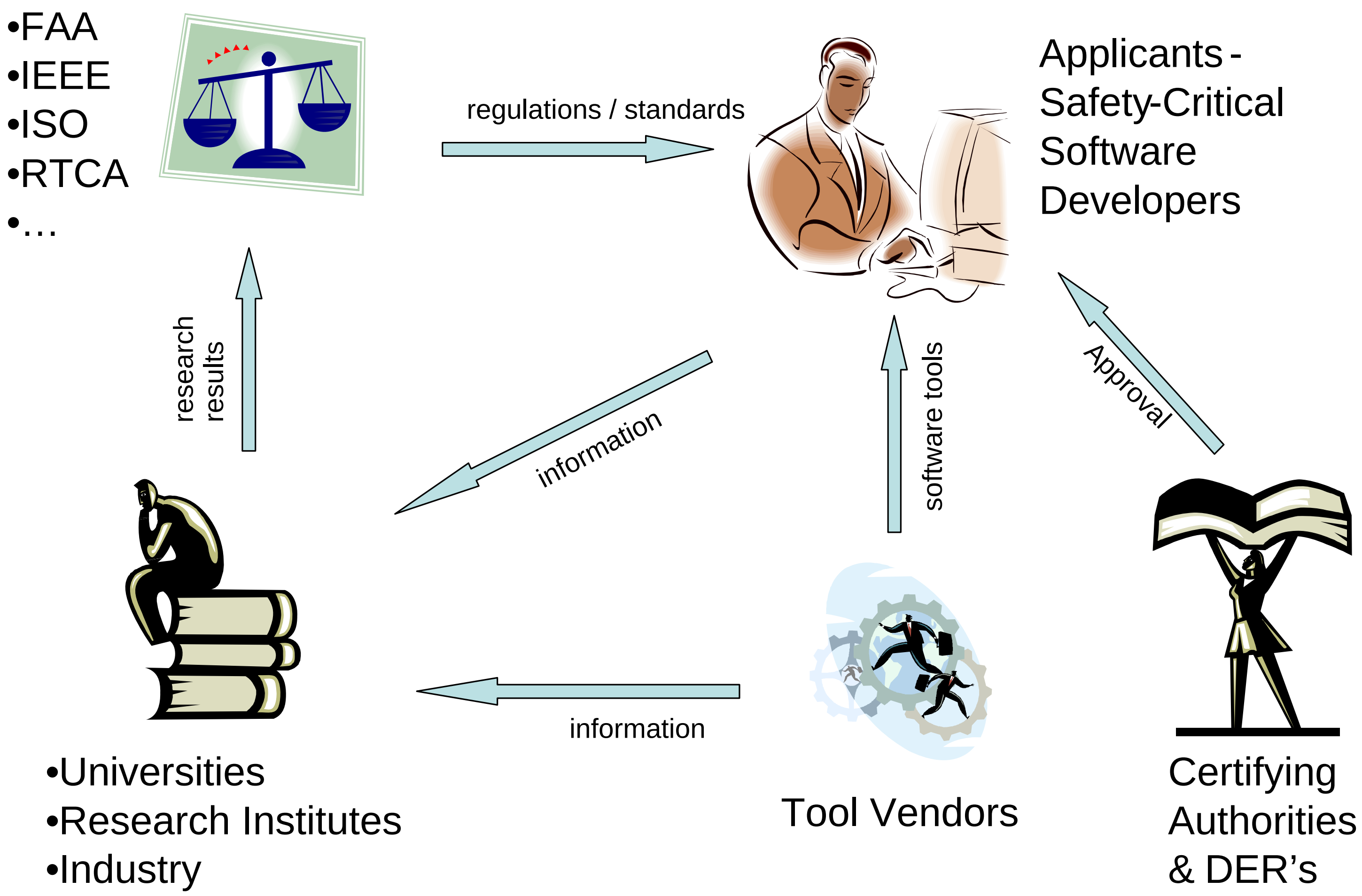
- Project Background
- **Research Approach**
- Development Tool Assessment
- Development Tool Qualification
- Experiments
- Software Tool Forum
- Research Findings
- Conclusions

Real Time Safety

Research Approach: FAA Contract via AACE

- A number of software tool manufacturers claim their tool will meet DO-178B criteria
- 2001 Aircraft Airworthiness Center of Excellence (AACE) research program solicitation
- Contract DTFA0301C00048: "Assessment of Development Tools for Safety Critical Real-Time Systems"
- Three phase research activity (Jan 2002-June 2005)
  - Phase One: Baseline and Taxonomy
  - Phase Two: Experiment and Feedback
  - Phase Three: Assessment and Guidelines
- The purpose of the project has been to assess the safety, quality, economic, efficiency benefits, and advantages/disadvantages of software tool suites used to develop software

Software Aspects of Certification Stakeholder Diagram



Real Time Safety

Research Approach: Objectives

- To **review literature** to establish a base for assessment of software development tools
- To **identify the development tools**, their impact on the software development lifecycle, and the qualification efforts
- To **conduct industry surveys** and analyze the industry/government feedback to define tool evaluation criteria
- To **create a taxonomy** and a set of criteria/guidelines for the tool selection and qualification
- To assess effort and defects during a case study **collecting experimental development process data**
- To analyze the data, edit the reports, and **disseminate the results**
- To **provide input for guidelines** on tools selection and qualification

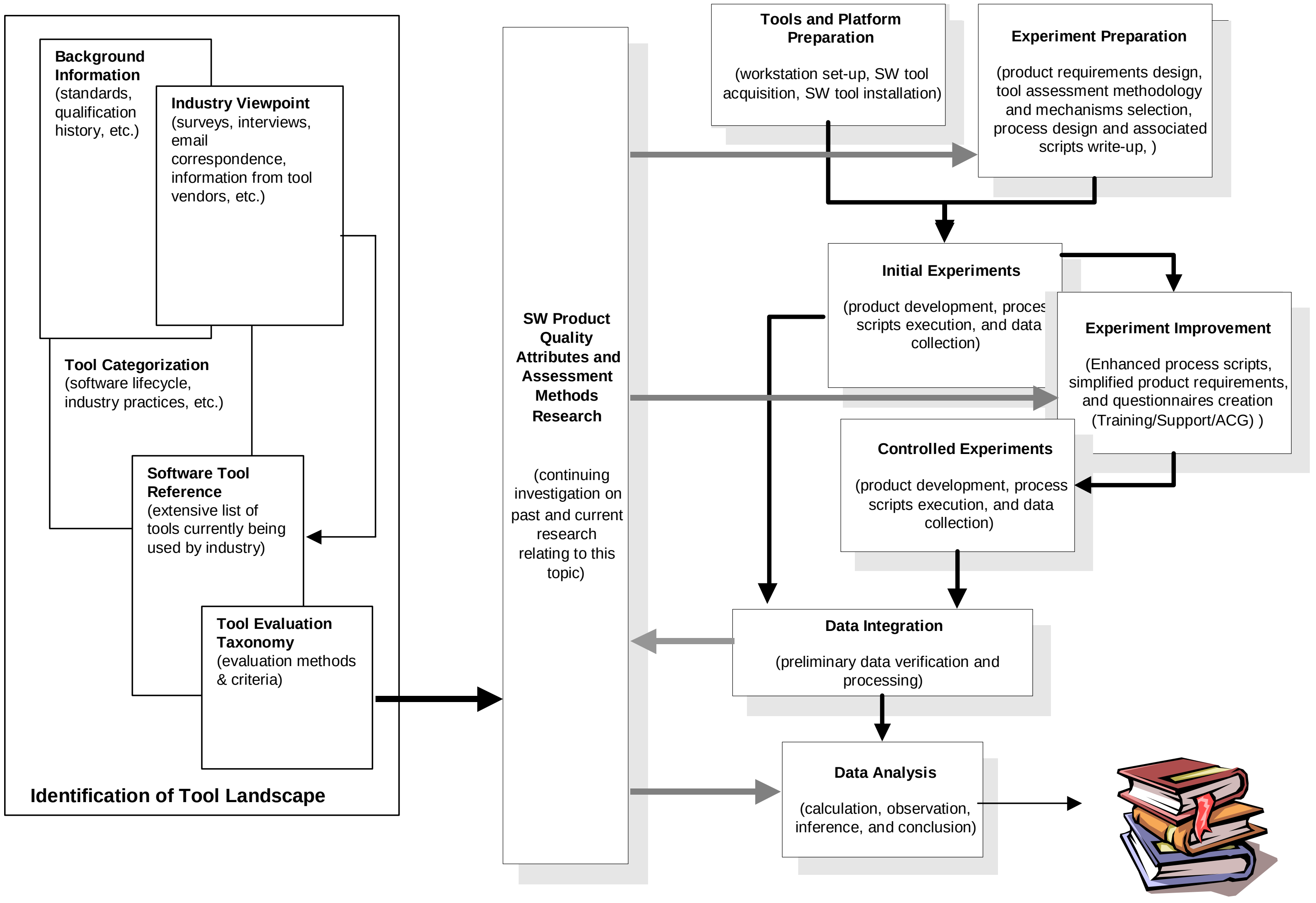
Real Time Safety

Research Approach: Problem Statement - Research Questions

- **Industry View**
  - Qualification Process and Approaches
  - Safe Use of Tools
  - Future Trends
- **Qualification**
  - What, Why and How?
  - Barriers and Factors
- **Quality Assessment**
  - Safety Critical Real-Time Specifics
  - Mechanisms and Methods of Assessment
  - Tool Support Impact
- **Tool Evaluation Taxonomy**
  - Tool Functionality and Categories
  - Evaluation Criteria

Research Approach: Project Workflow

Real Time Safet



Background Information (standards, qualification history, etc.)
Industry Viewpoint (surveys, interviews, email correspondence, information from tool vendors, etc.)
Tool Categorization (software lifecycle, industry practices, etc.)
Software Tool Reference (extensive list of tools currently being used by industry)
Tool Evaluation Taxonomy (evaluation methods & criteria)
Identification of Tool Landscape
SW Product Quality Attributes and Assessment Methods Research
(continuing investigation on past and current research relating to this topic)
Tools and Platform Preparation
(workstation set-up, SW tool acquisition, SW tool installation)
Experiment Preparation
(product requirements design, tool assessment methodology and mechanisms selection, process design and associated scripts write-up,)
Initial Experiments
(product development, proces scripts execution, and data collection)
Experiment Improvement
(Enhanced process scripts, simplified product requirements, and questionnaires creation (Training/Support/ACG))
Controlled Experiments
(product development, process scripts execution, and data collection)
Data Integration
(preliminary data verification and processing)
Data Analysis
(calculation, observation, inference, and conclusion)

Real Time Safety

Outline

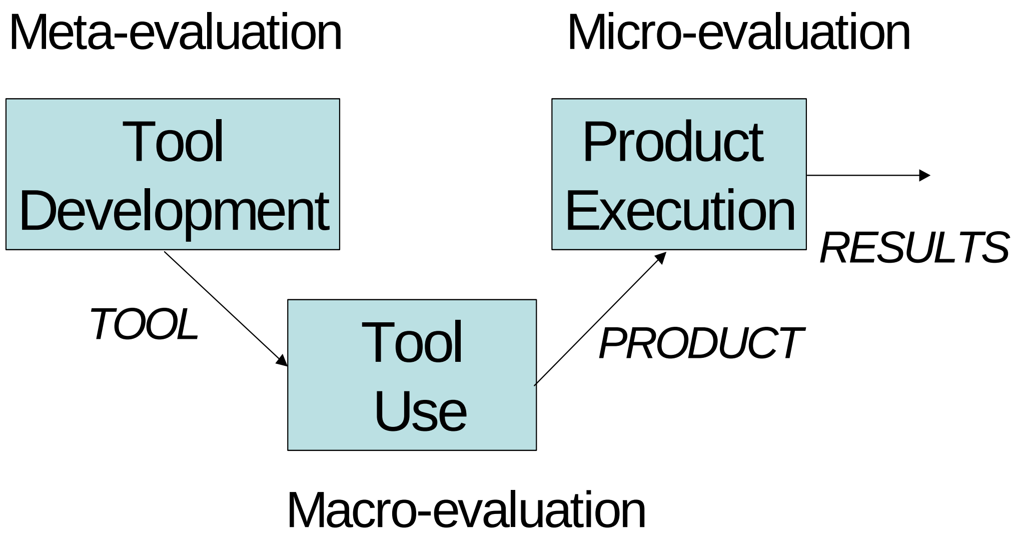
- Project Background
- Research Approach
- **Development Tool Assessment**
- Development Tool Qualification
- Experiments
- Software Tool Forum
- Research Findings
- Conclusions

Real Time Safety

Assessment: Goals

- To help providing sufficient information to support the qualification of a tool
- To develop a set of criteria and methods to assess (measure) the quality of the tool: how well provides its function?

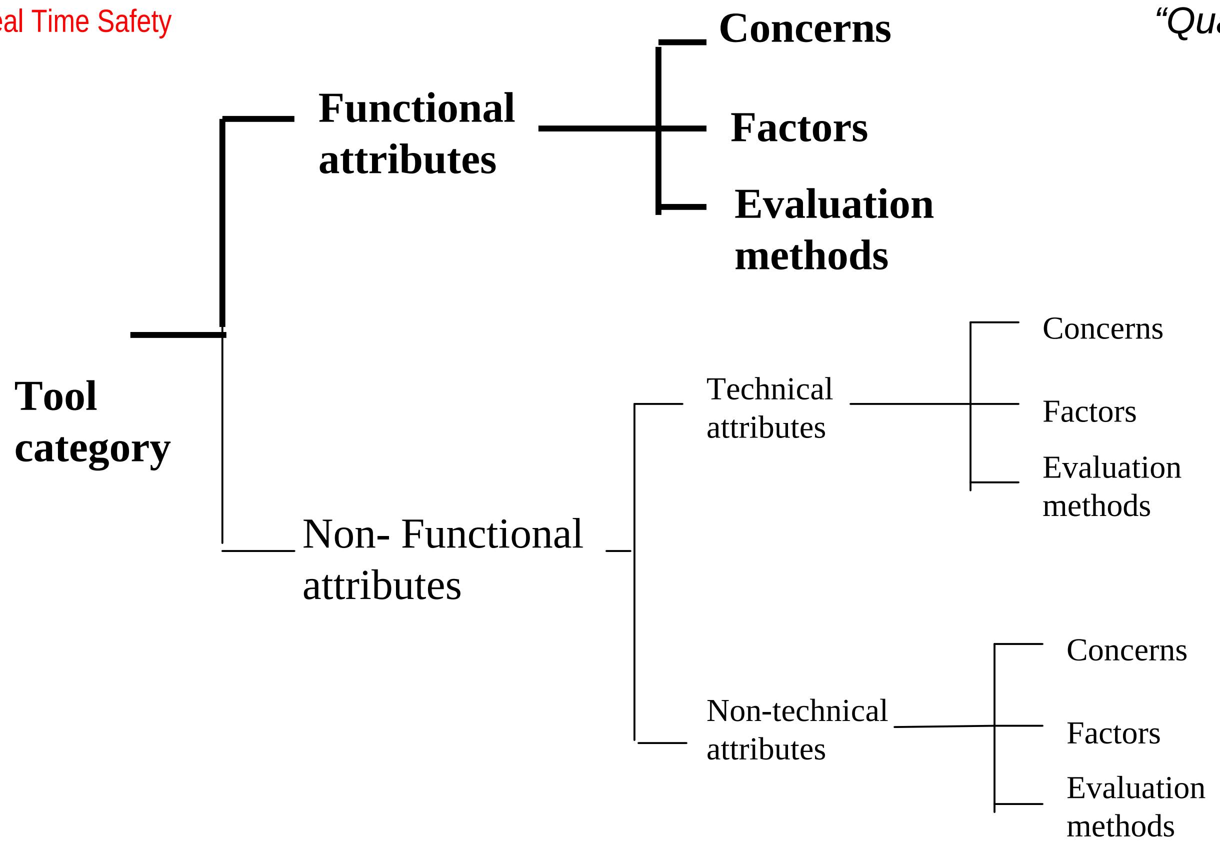
- Two approaches:
  - o Formal **qualification-oriented** evaluation of functionalities
  - o Informal **utilization-oriented** hands-on evaluation of the tool in operation



Real Time Safety

Assessment: Tool Evaluation Taxonomy

Source: TR CMU/SEI-95-TR-021
“Quality Attributes”, Barbacci, M. at al



Non-functional:

- Technical: *dependability, performance, security*
- Non-technical: *support, cost, vendor viability, training*

Functional:

- *determinism, correctness, robustness, traceability, standards conformance*

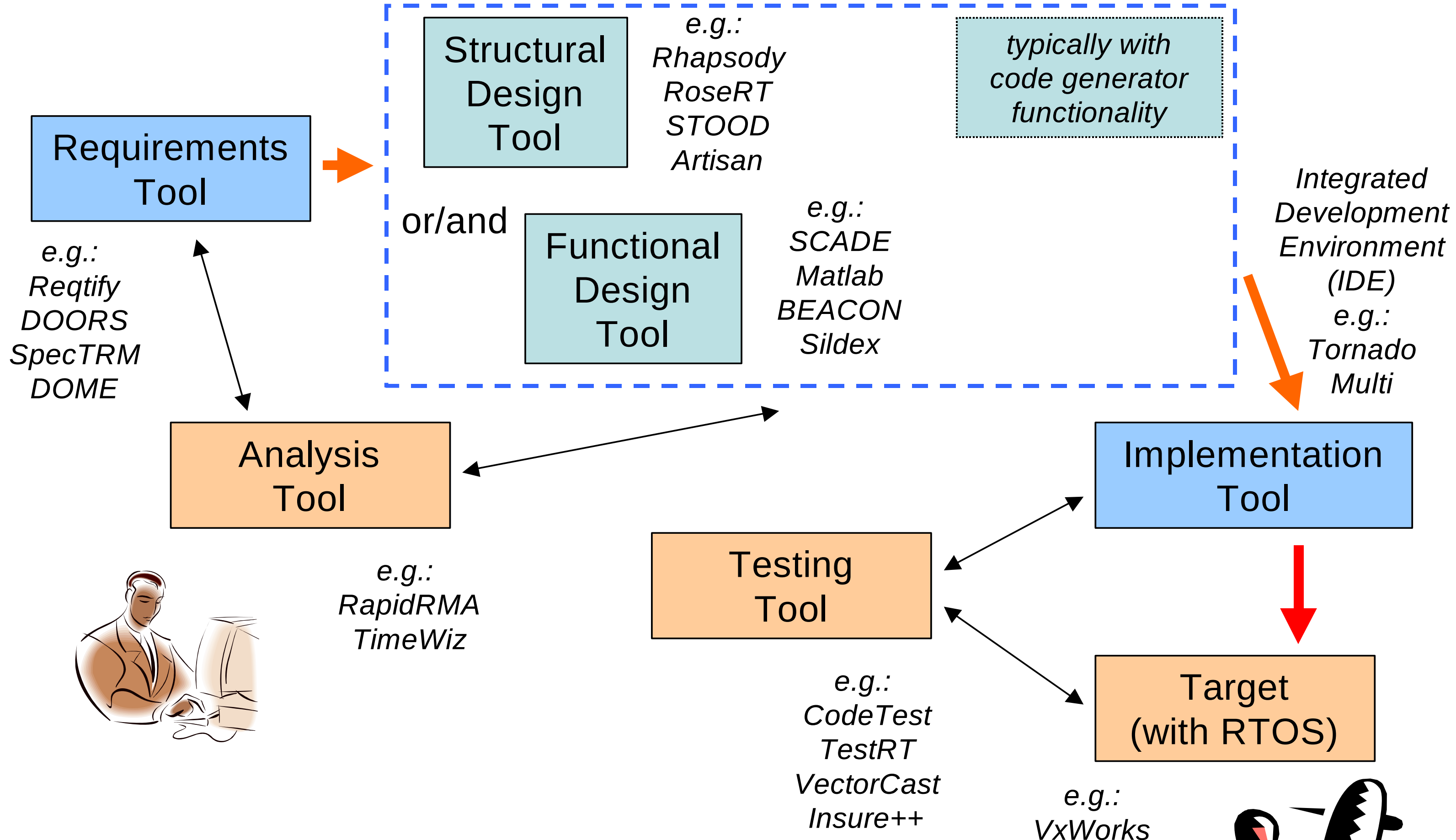
Real Time Safety

Assessment: Functional Attributes

concerns	factors	methods
Determinism	Tools' architecture Predictability Conversion and code generation Subset of language used in product	Architecture assessment (meta) Tool use (macro) Algorithm inspection (meta) Code inspection (macro)
Robustness	Partition integrity Boundary conditions Architecture / Coupling	Code inspection (macro) Code testing (micro) Design review (meta)
Traceability	Product's architecture Product's code / coding rules	Model inspection (macro) Code inspection (macro)
Correctness	Conversion and code generation Subset of language used Syntax and formality Real-time management Language representation rules	Algorithm inspection (meta) Code inspection (macro) Formal techniques (macro) Timing evaluation (micro) Formal methods (meta)
Standards Conformance	Design Coding Behavioral	Model inspection (macro) Code inspection (macro) System testing (micro)

Real Time Safety

Assessment: Tools Landscape

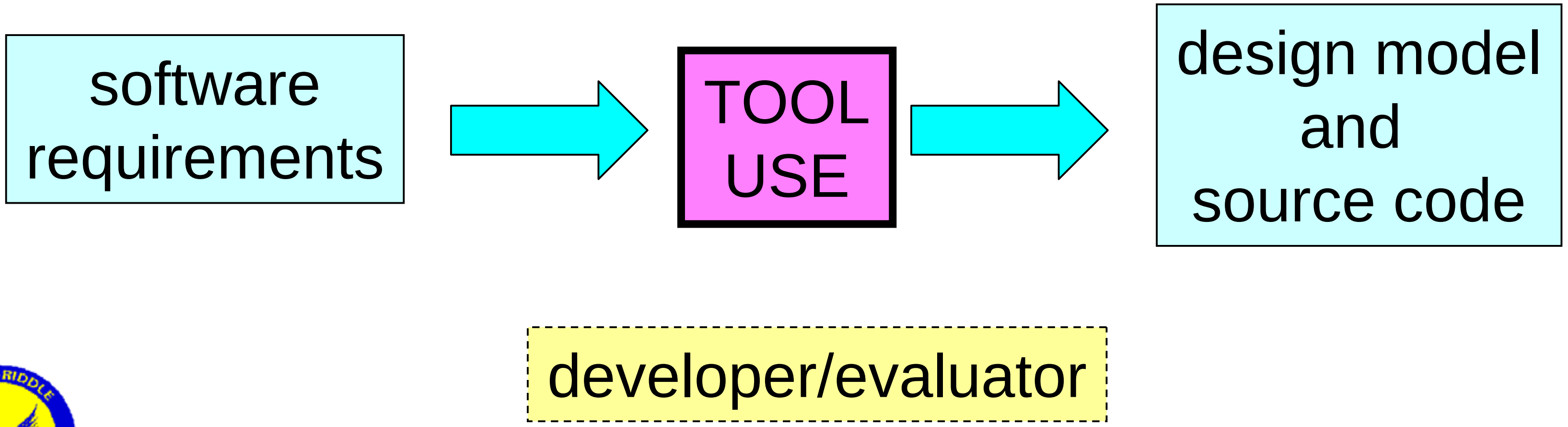


Requirements Tool
e.g.: Reqtify DOORS SpecTRM DOME
or/and
Structural Design Tool
e.g.: Rhapsody RoseRT STOOD Artisan
typically with code generator functionality
Functional Design Tool
e.g.: SCADE Matlab BEACON Sildex
Integrated Development Environment (IDE) e.g.: Tornado Multi
Analysis Tool
e.g.: RapidRMA TimeWiz
Implementation Tool
Testing Tool
e.g.: CodeTest TestRT VectorCast Insure++
Target (with RTOS)
e.g.: VxWorks QNX OSE Integrity LynxOS

Tool Categories

Assessment: Framework

- Data required for evaluation (each category needs data items to collect criteria/metrics):
  - o tool data
  - o model data
  - o source code data



Assessment: Data Items

Tool

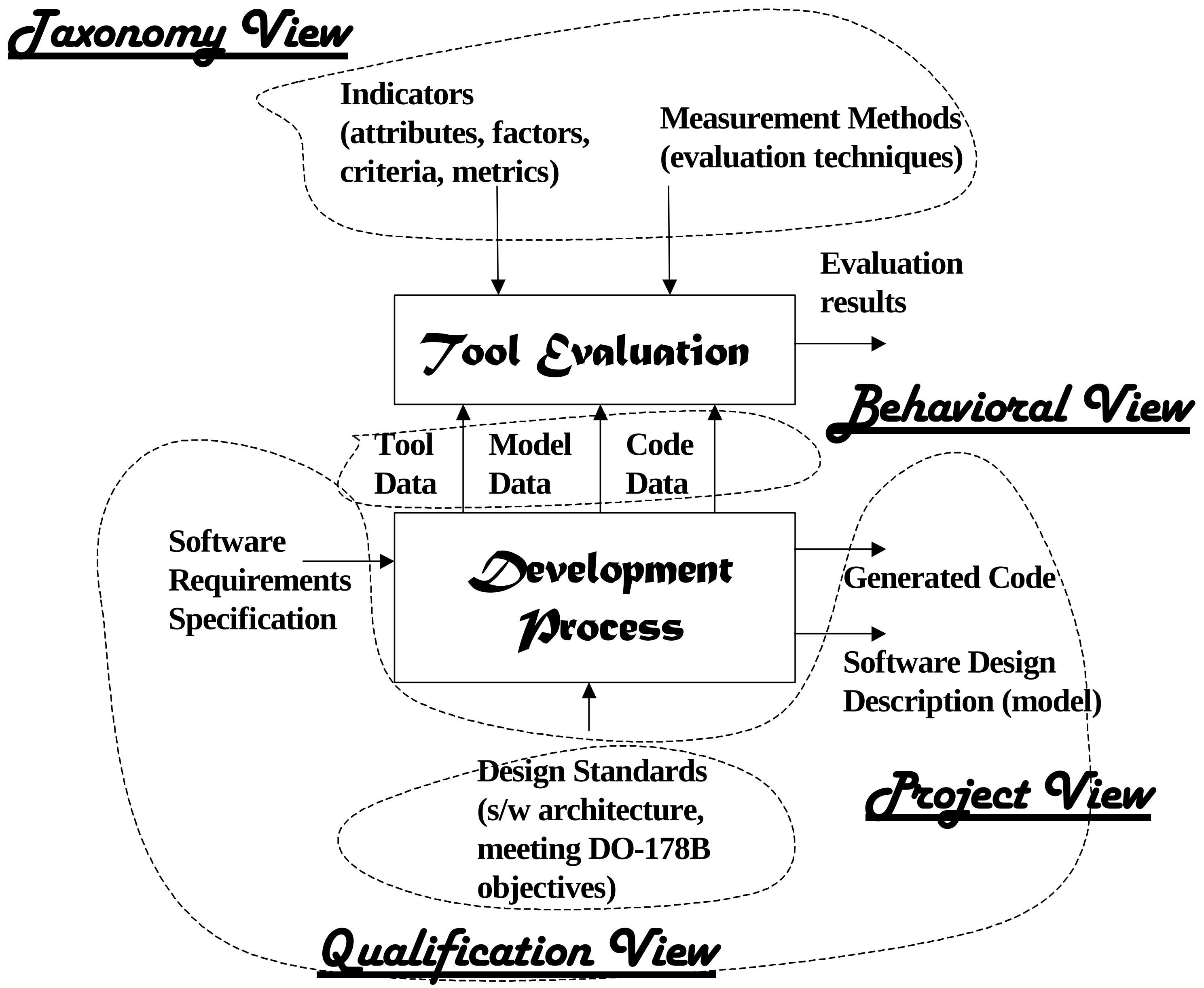
- Operational Profile
- Environment Specification
- Functional Characteristics
- Methodology and Notation Primitives
- Support for Expressing Timing and Safety Properties
- Requirement Document
- Design Document
- Test Scenarios and Results

Model

- Software Requirement Document
- Software Safety Properties
- Size of Model (blocks, modules, interfaces)
- Software Operational Scenarios
- Expected Timing Characteristics
- Expected Behavioral Model (states, sequences)

Code

- Partial vs. full code generation
- Code properties (language, subsets)
- Code parameters (LOC, quantitative measures)
- Readability (names, indentation, comments)
- Partitioning (separation of data, mutual exclusion)
- Timing
- Executable size (memory allocation)


Taxonomy View
Indicators (attributes, factors, criteria, metrics)
Measurement Methods (evaluation techniques)
Evaluation results
Tool Evaluation
Behavioral View
Tool Data
Model Data
Code Data
Software Requirements Specification
Development Process
Generated Code
Software Design Description (model)
Design Standards (s/w architecture, meeting DO-178B objectives)
Project View
Qualification View

Real Time Safety

Outline

- Project Background
- Research Approach
- Development Tool Assessment
- **Development Tool Qualification**
- Experiments
- Software Tool Forum
- Research Findings
- Conclusions

Real Time Safety

Qualification: Objective

- Establish assurance that the software requirements, as submitted to the tool and developer, are correctly and completely transformed into the generated source code
- Qualification process involves the user to perform an audit of the tool, with an active cooperation of the tool vendor, who needs to show that:
  - Tool requirements are properly documented
  - Tool have appropriate configuration management
  - Tool has been satisfactorily tested against its requirements
- Procedures controlling the use of tools (environment, constraints, limitations, version control) are as important as tool itself

Real Time Safety

Qualification: Process

- Tool qualification is an integral component of a specific certification program referenced within the Plan for Software Aspects of Certification (PSAC) and Software Accomplishment Summary (SAS) of the original certification project
- For development tools the required documents are:
  - Tool Qualification Plan (TQP)
  - Tool Operational Requirements (TOR)
  - Tool Accomplishment Summary (TAS)
- Other data, required for review, include Tool Configuration Management Index, Tool Development Data, Tool Verification Records, Tool Quality Assurance Records, Tool Configuration Management Records, etc.

Outline

Real Time Safety

Experiments: Selected Tools

- For detailed analysis we selected design tools, with a **code generator** functionality, identifying two groups of different modeling paradigm:
- Structural/Discrete Models (UML-oriented):
  - Rose Real Time / Rational http://www.rational.com/
  - Rhapsody / iLogix http://www.ilogix.com/
  - Real Time Studio Professional / Artisan http://www.artisansw.com/
  - STOOD / TNI-Valiosys http://www.tni-valiosys.com/
- Functional/Continuous Models (Block-oriented):
  - SCADE / Esterel http://www.esterel-technologies.com/
  - Sildex / TNI-Valiosys http://www.tni-valiosys.com/
  - Simulink/RTW / Mathworks / RTW http://www.mathworks.com/
  - Tau SDL Suite / Telelogic http://www.telelogic.com

Real Time Safety

Experiments: Description

- **Process**:
  - o Preparation: project and tool familiarization
  - o Model development, code generation, and implementation
  - o Data collection
  - o Post-mortem
- **System**:
  - o <u>Preliminary Experiment</u> (4 tools, 4 developers):
    - ➢ flight data collection with simple processing (averaging, time-stamping) and displaying results on the terminal under user control
  - o <u>Controlled Experiment</u> (6 tools, 14 developers):
    - ➢ hair-dryer simulator (training);
    - ➢ microwave oven simulator (data collection)

Real Time Safety

Experiments: Criteria / Sources

- {**178B**} RTCA Inc. (1992) *Software Considerations in Aiborne Systems and Equipment Certification*, Report RTCA/DO-178B, Washington, DC
- {**ISO**} ISO/IEC 14102-1995 (1995) *Information Technology – Guideline for the Evaluation and Selection of CASE Tools*, ISO, Geneva, Switzerland, 15 November 1995
- {**FAA**} U.S Department of Transportation, Federal Aviation Administration (1991) *Software Quality Metrics*, Report DOT/FAA CT-91/1, 1991
- {**VTT**} Ihme T, Kumara P, Suihkonen K, Holsti N, Paakko M (1998) *Developing Application Frameworks for Mission-Critical Software: Using Space Applications as an Example*, Research Notes 1933, Technical Research Centre of Finland, Espoo, 1998
- {**BSC**} Wichmann B (1999) *Guidance for the Adoption of Tools for Use in Safety Related Software Development*, Report, British Computer Society, March 1999

Real Time Safety

Experiments: Criteria Selection

Criteria	survey	178B	ISO	FAA	VTT	BCS
Consistency		*				
Efficiency			*		*	
Functionality	*		*		*	
Maintainability			*	*		
Modifiability		*		*		
Portability			*	*		
Reliability	*	*	*	*		
Robustness		*				
Traceability		*				*
Usability			*	*		

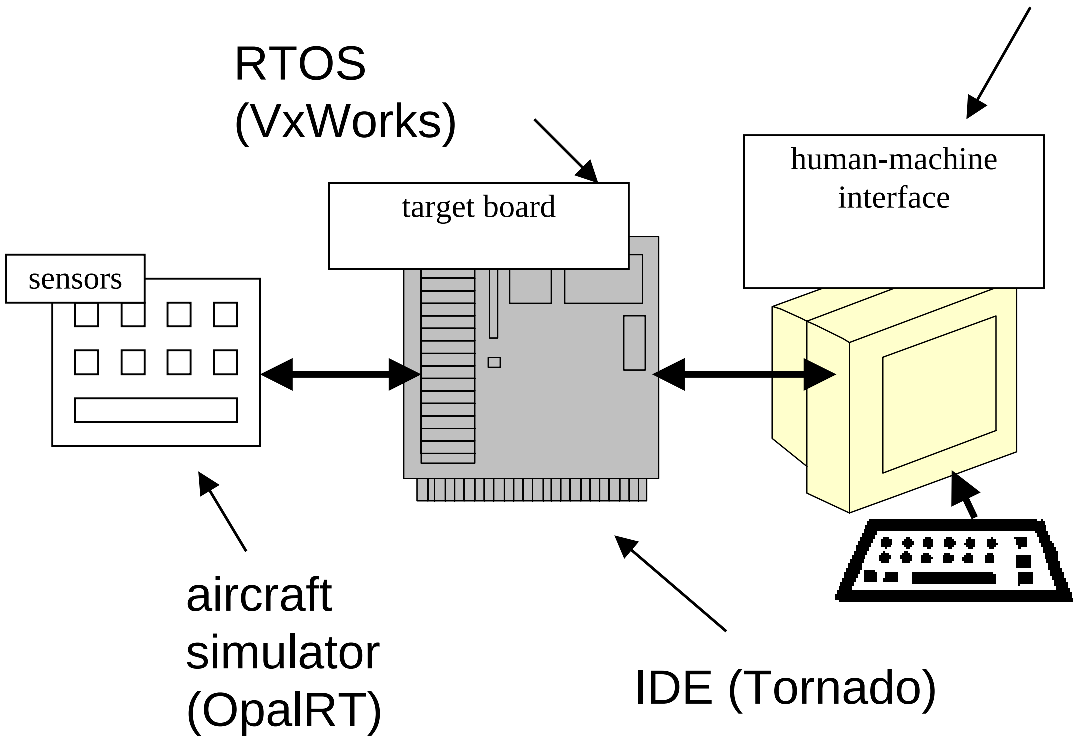
- **Efficiency**: code size
- **Functionality**: questionnaire feedback
- **Traceability**: manual tracking between model and code
- **Usability**: developers effort (PSP)

Preliminary Experiment

Traceability assessment matching:
requirements ⇔ design ⇔ code

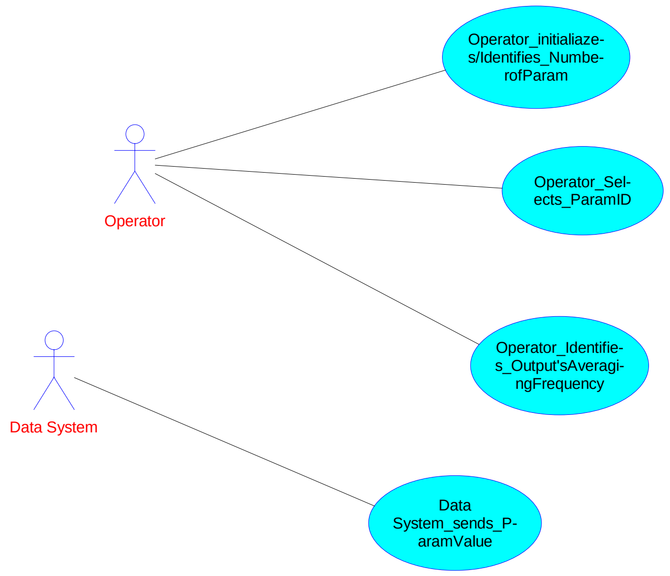
- Purpose: to establish a **base for assessment of software development tools**
- Personal Software Process used to capture effort data
- Process improvement

Development tools:
Matlab/Simulink/RTW,
SCADE, Sildex, Artisan

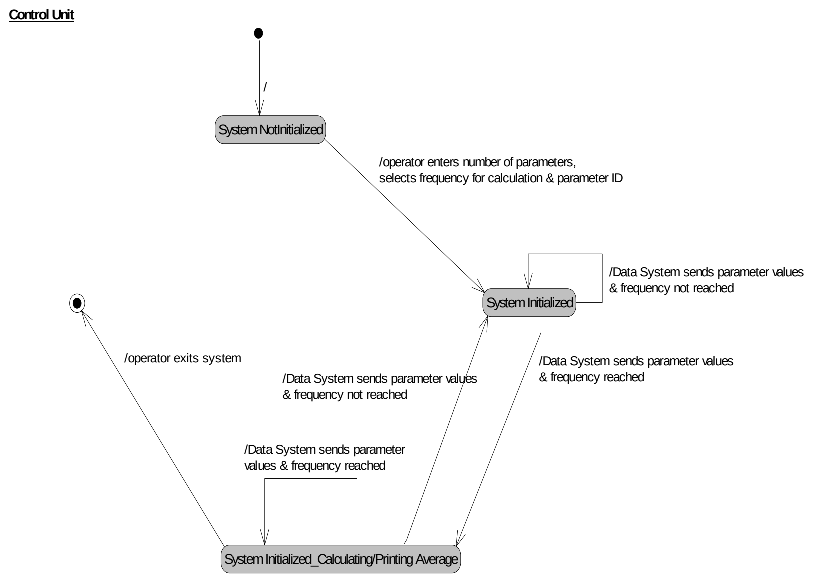


Real Time Safety

Example: Flight Data Acquisition Modeling



Operator_initialiaze-s/Identifies_Numbe-rofParam
Operator_Sel-ects_ParamID
Operator
Operator_Identifie-s_Output'sAveragi-ngFrequency
Data System
Data System_sends_P-aramValue



Control Unit
/
System NotInitialized
/operator enters number of parameters, selects frequency for calculation & parameter ID
/Data System sends parameter values & frequency not reached
System Initialized
/operator exits system
/Data System sends parameter values & frequency not reached
/Data System sends parameter values & frequency reached
/Data System sends parameter values & frequency reached
System Initialized_Calculating/Printing Average

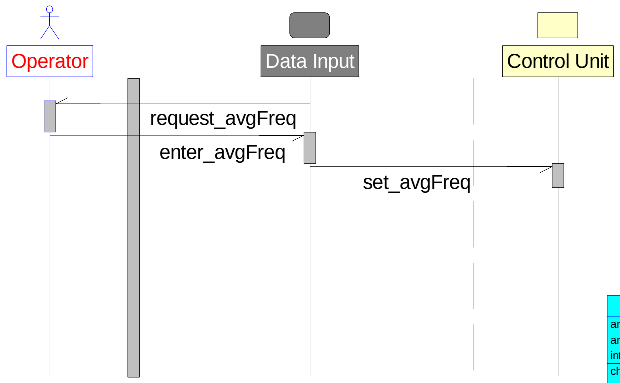
Operator_Identifies_Output'sAveragingFrequency

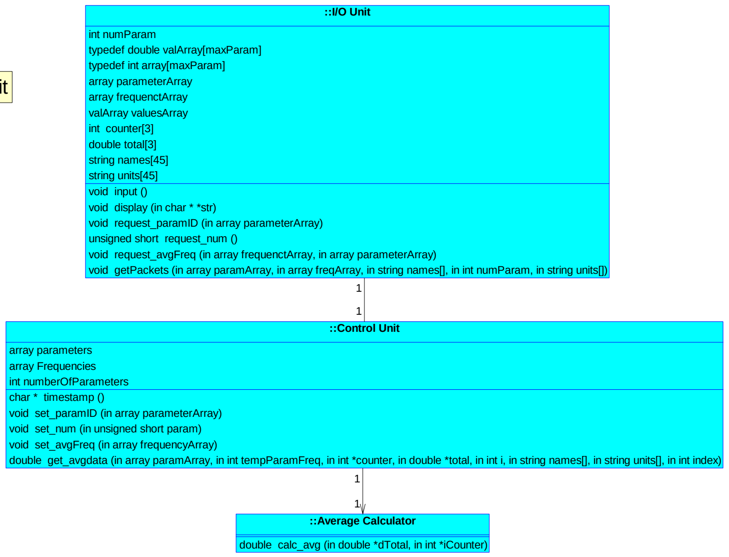
Description

Data Input requests averaging frequency

Operator enters the parameter ID

Control Unit sets the parameter ID



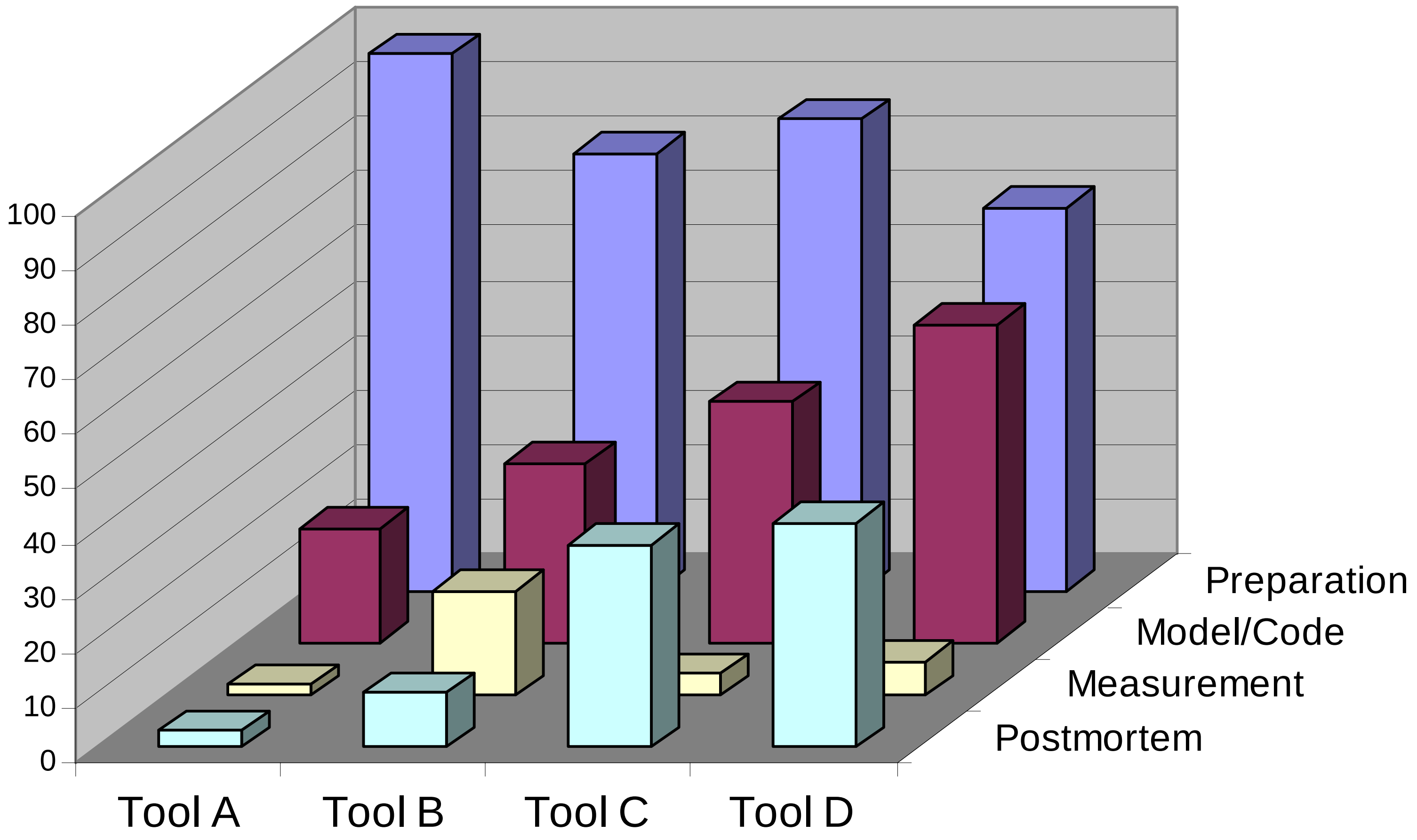


```
::I/O Unit
int numParam
typedef double valArray[maxParam]
typedef int array[maxParam]
array parameterArray
array frequenctArray
valArray valuesArray
int  counter[3]
double total[3]
string names[45]
string units[45]
void  input ()
void  display (in char * *str)
void  request_paramID (in array parameterArray)
unsigned short  request_num ()
void  request_avgFreq (in array frequenctArray, in array parameterArray)
void  getPackets (in array paramArray, in array freqArray, in string names[], in int numParam, in string units[])
```

```
::Control Unit
array parameters
array Frequencies
int numberOfParameters
char *  timestamp ()
void  set_paramID (in array parameterArray)
void  set_num (in unsigned short param)
void  set_avgFreq (in array frequencyArray)
double  get_avgdata (in array paramArray, in int tempParamFreq, in int *counter, in double *total, in int i, in string names[], in string units[], in int index)
```

```
::Average Calculator
double  calc_avg (in double *dTotal, in int *iCounter)
```

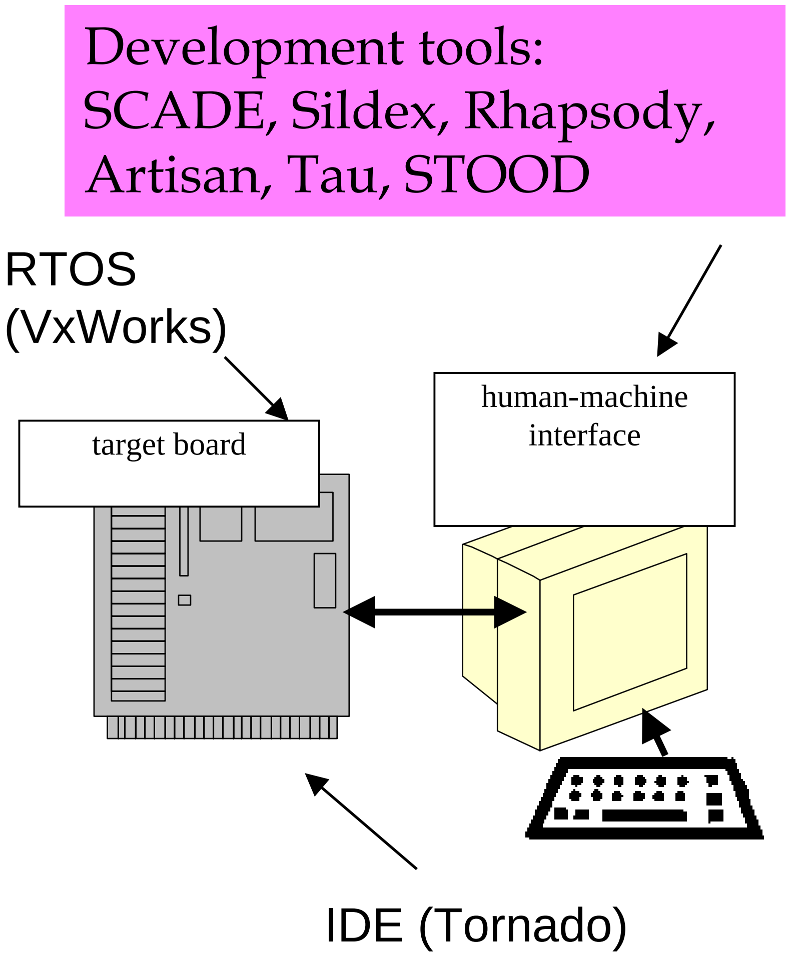
Preliminary Experiment: Usability



Real Time Safety

Controlled Experiment

- Purpose: to collect data supporting **assessment of software development tools**
- Personal Software Process used to capture effort data
- Questionnaire used to collect qualitative data
- Two-phase approach
  - Training model: hair-dryer *(4 requirements)*
  - Experimental model: microwave *(10 requirements)*


Development tools: SCADE, Sildex, Rhapsody, Artisan, Tau, STOOD
RTOS (VxWorks)
target board
human-machine interface
IDE (Tornado)

Real Time Safety

Requirements microwave oven:

hair-dryer:

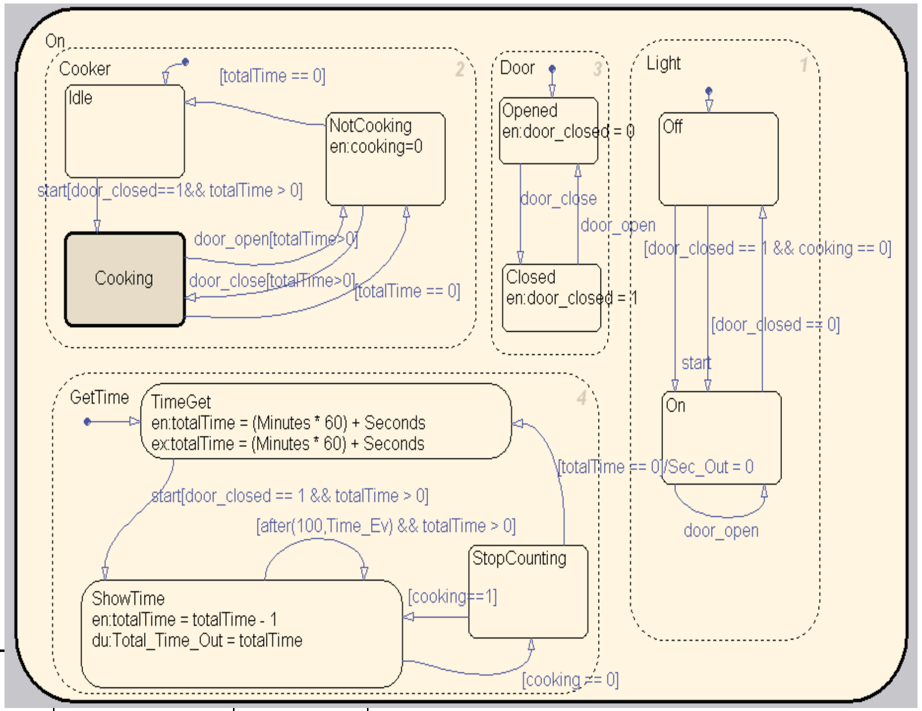
1. *The system shall allow user to select motor speed (off, low, or high).*
2. *The system shall apply power to motor depending on the selected speed setting.*
3. *The system shall cycle the heater (30 seconds on and 30 seconds off) when in low- and high- speed modes.*
4. *The system display shall show the selected speed, heater status and the count down time when the heater is on.*

microwave oven:

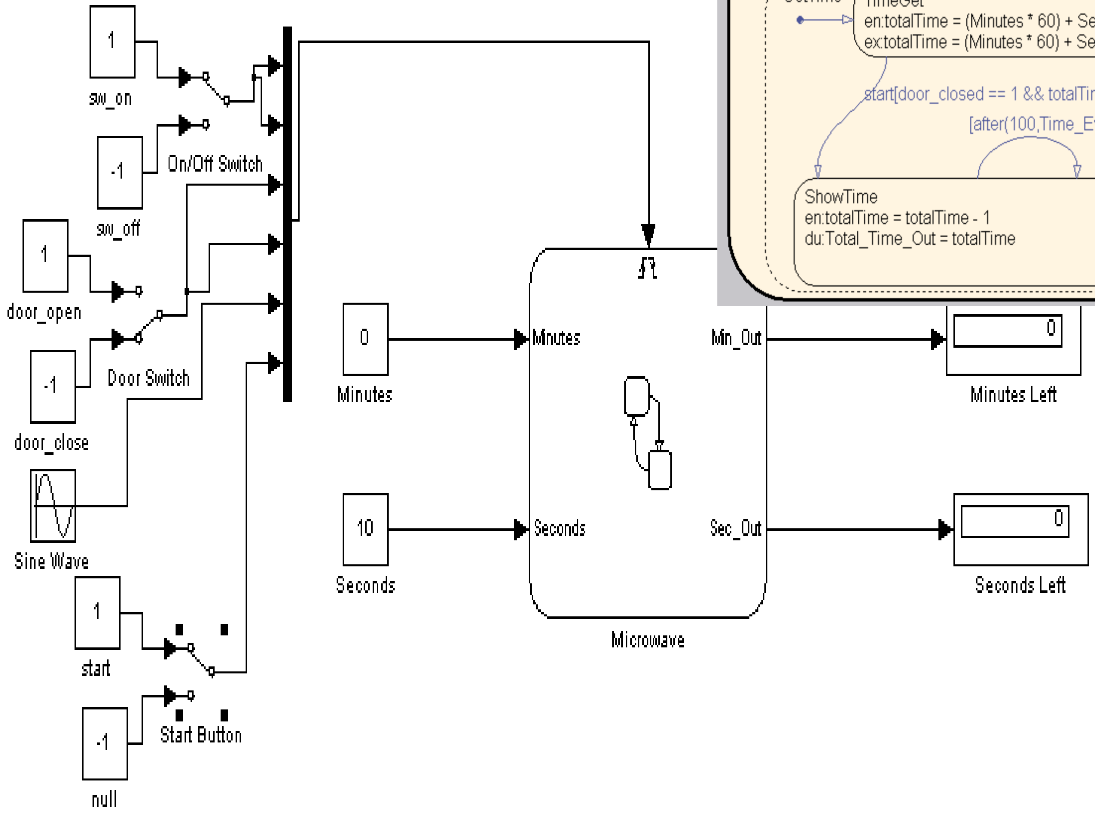
1. *The oven shall allow user to set the cooking time in minutes and seconds (from default 00:00 to 59:59).*
2. *The oven shall allow user to set the power level (in the range from default 1 to 5).*
3. *The start of cooking shall initiate on explicit user request.*
4. *When the cooking starts, the oven shall turn on the light and the rotisserie motor for the specified time period.*
5. *When the cooking starts the oven shall cycle the microwave emitter on and off: the power level of 5 means that the emitter is on all the time, the power level of 1 means that the emitter is on only 1/5th of the time.*
6. *The oven shall display the remaining time of the cooking and the power level.*
7. *When the time period expires, the audible sound shall be generated and the light, motor, and emitter shall be turned off.*
8. *The oven shall turn on the emitter and the motor only when the door is closed.*
9. *The oven shall turn on the light always when the door is open.*
10. *The oven shall allow the user to reset at any time (to the default values)*

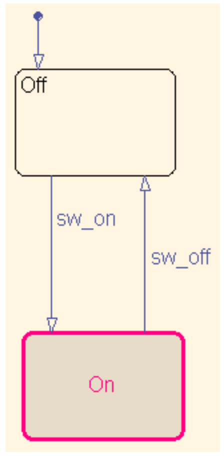
Real Time Safety

Example: Microwave Modeling



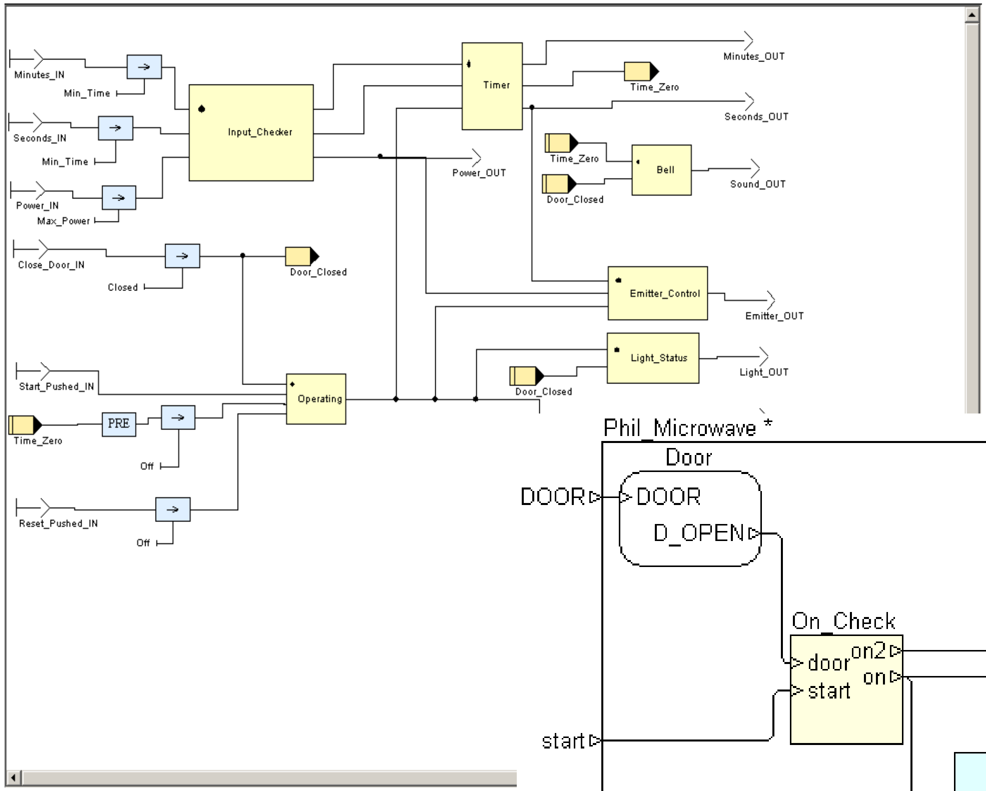
On
Cooker
Idle
[totalTime == 0]
NotCooking
en:cooking=0
start[door_closed==1&& totalTime > 0]
Cooking
door_open[totalTime>0]
door_close[totalTime>0]
[totalTime == 0]
Door
Opened
en:door_closed = 0
door_close
door_open
Closed
en:door_closed = 1
Light
Off
[door_closed == 1 && cooking == 0]
[door_closed == 0]
start
On
[totalTime == 0]/Sec_Out = 0
door_open
GetTime
TimeGet
en:totalTime = (Minutes * 60) + Seconds
ex:totalTime = (Minutes * 60) + Seconds
start[door_closed == 1 && totalTime > 0]
[after(100,Time_Ev) && totalTime > 0]
StopCounting
ShowTime
en:totalTime = totalTime - 1
du:Total_Time_Out = totalTime
[cooking==1]
[cooking == 0]

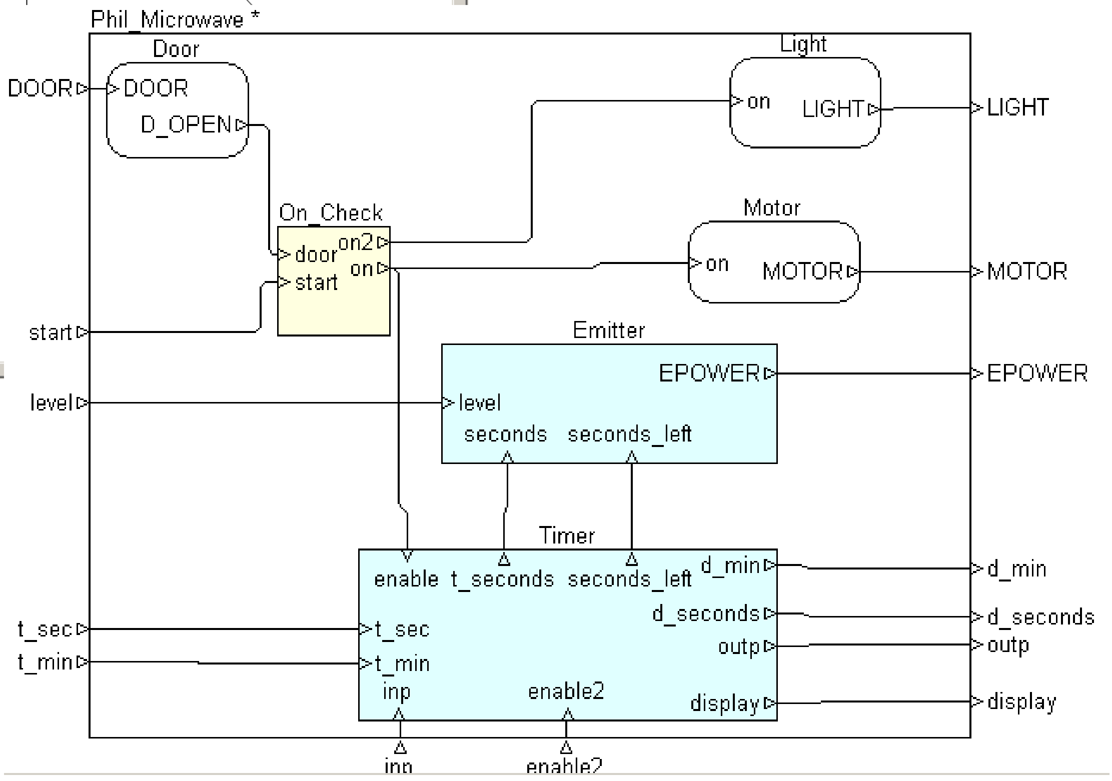




Real Time Safety

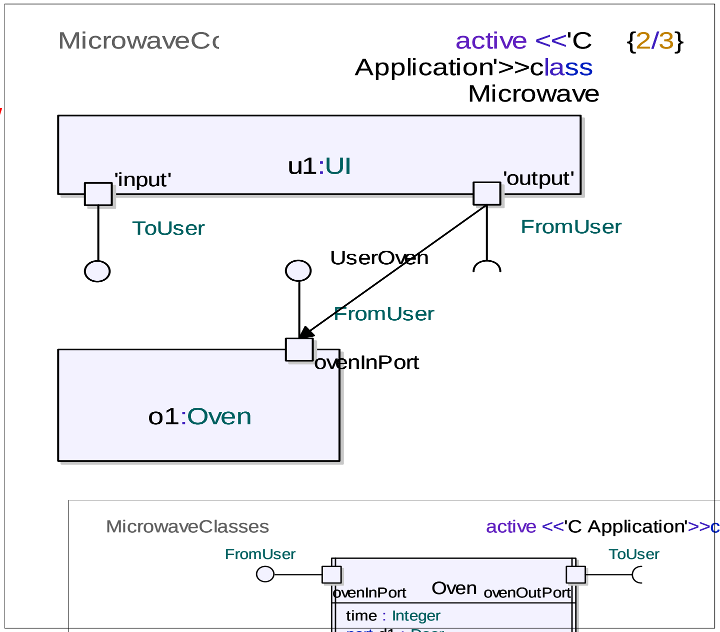
Example: Microwave Modeling

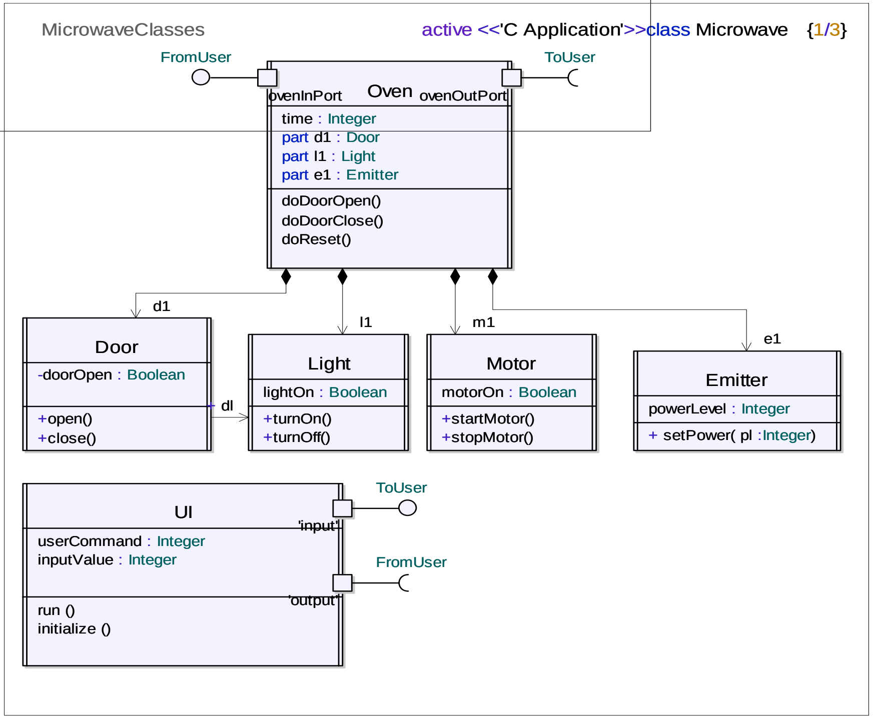


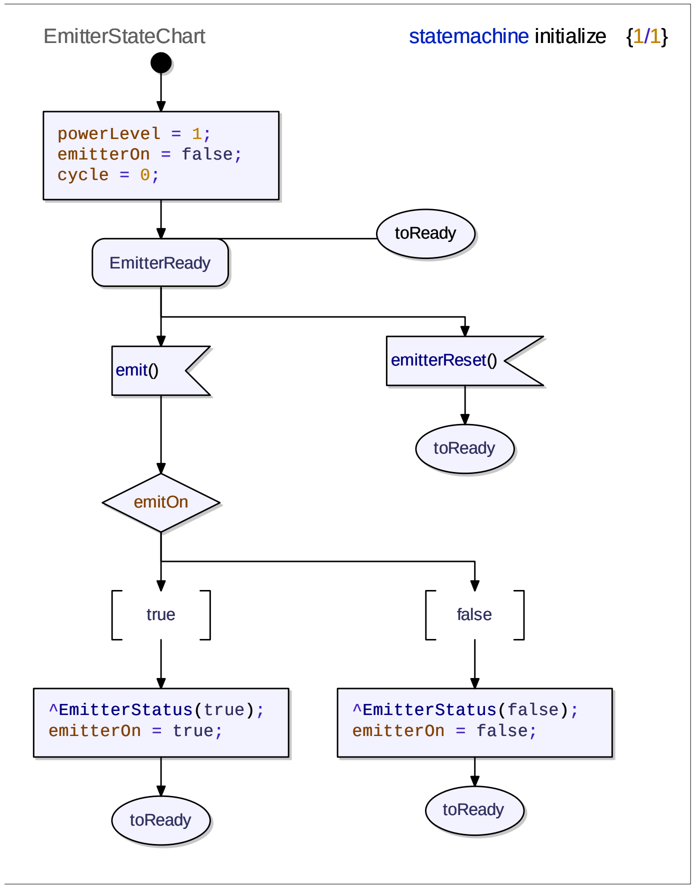


Real Time Safety

Example: Microwave Modeling

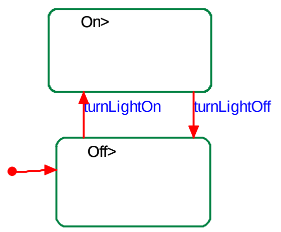


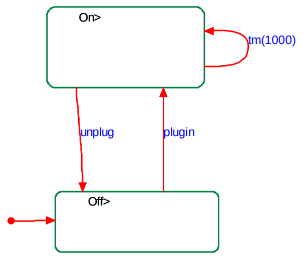


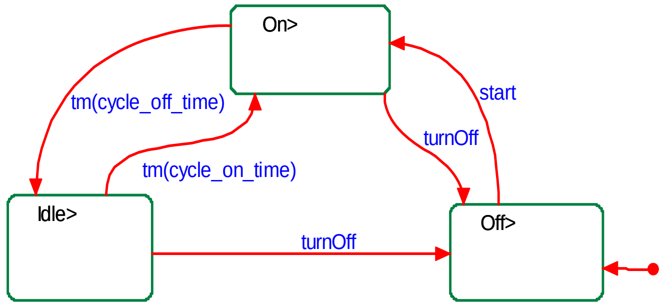


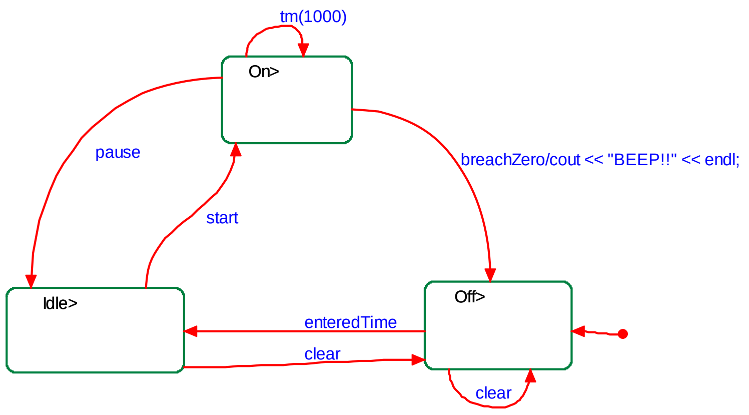
Example: Microwave Modeling

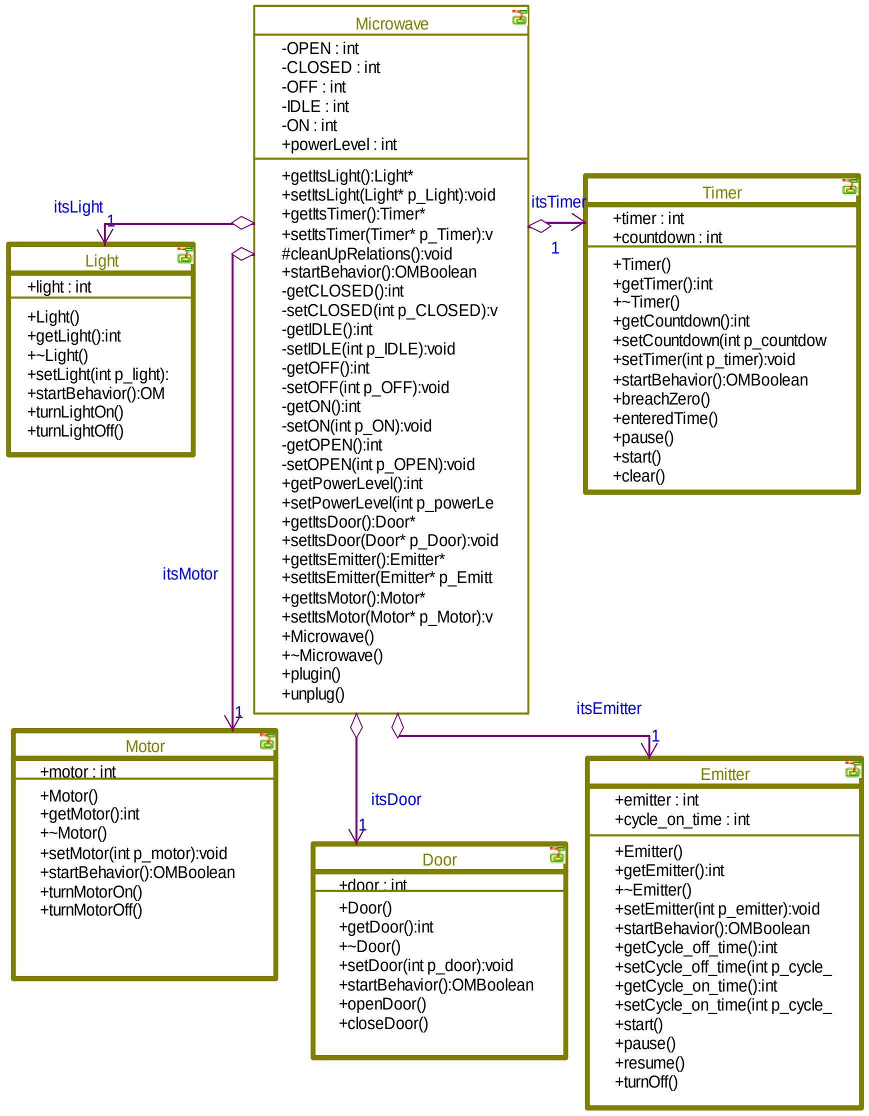
Real Time Safety

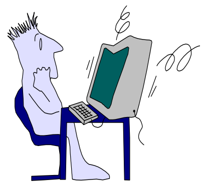






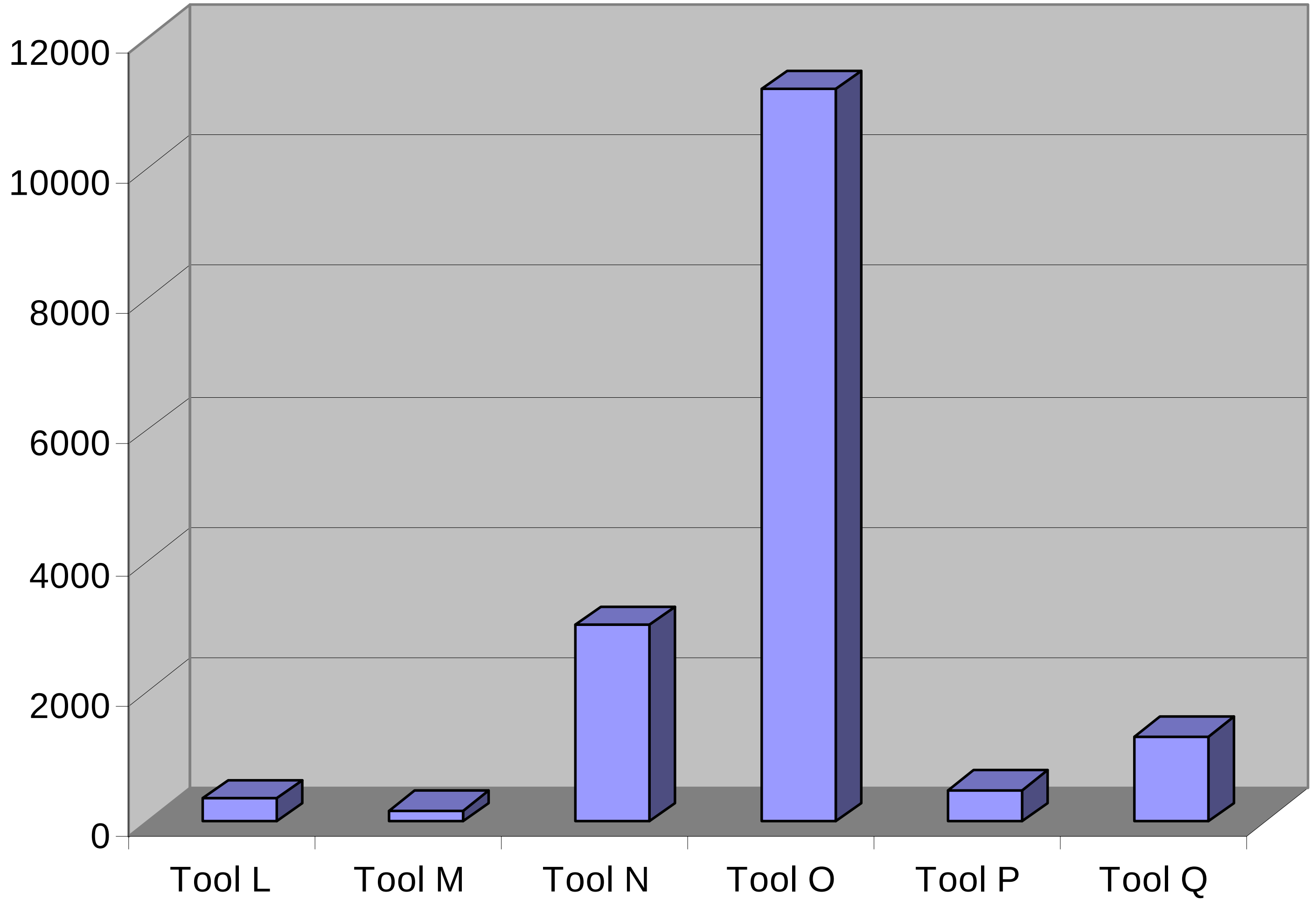






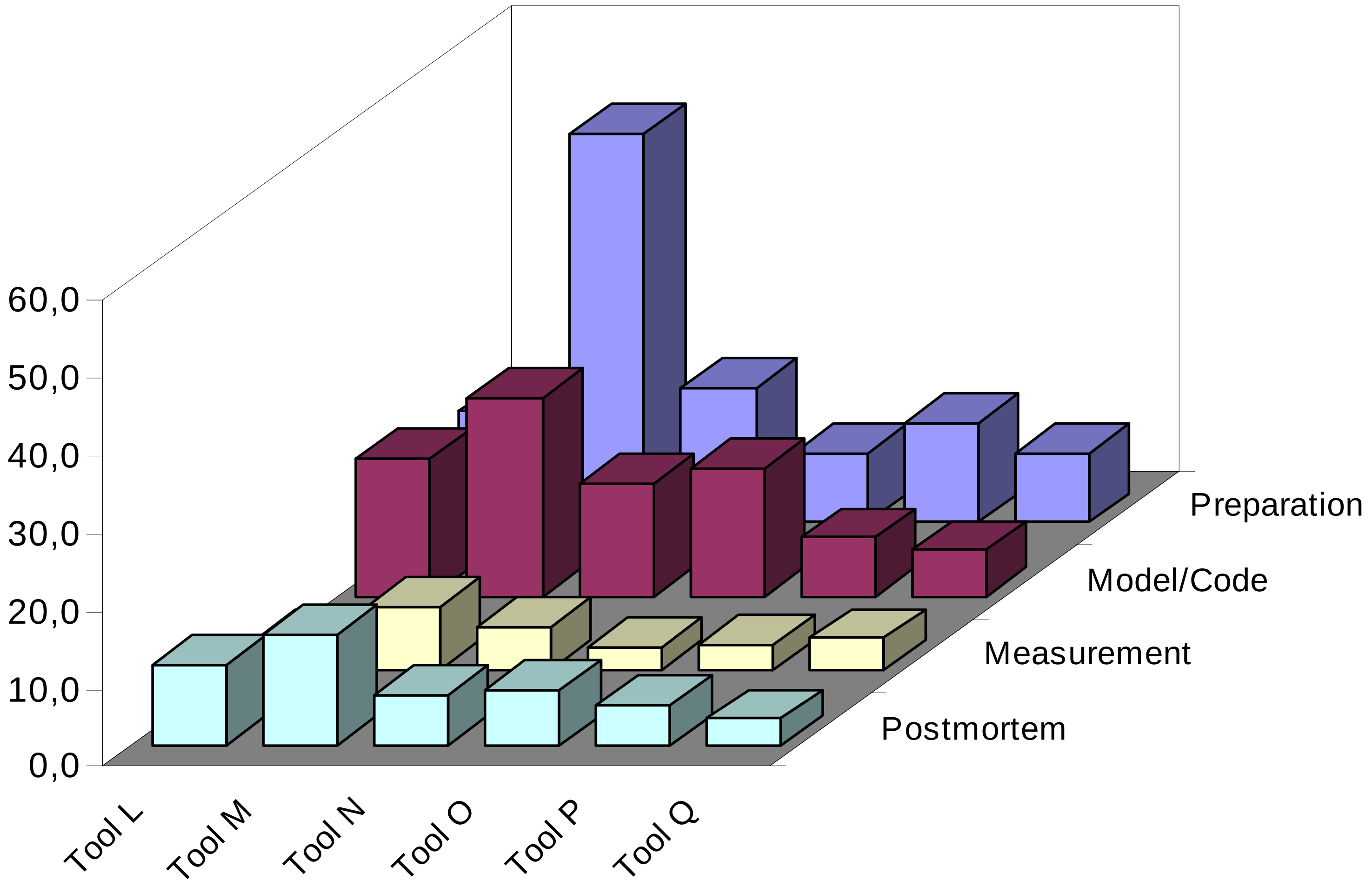
Real Time Safety

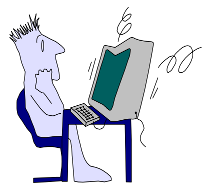
Controlled Experiment: Efficiency



Real Time Safety

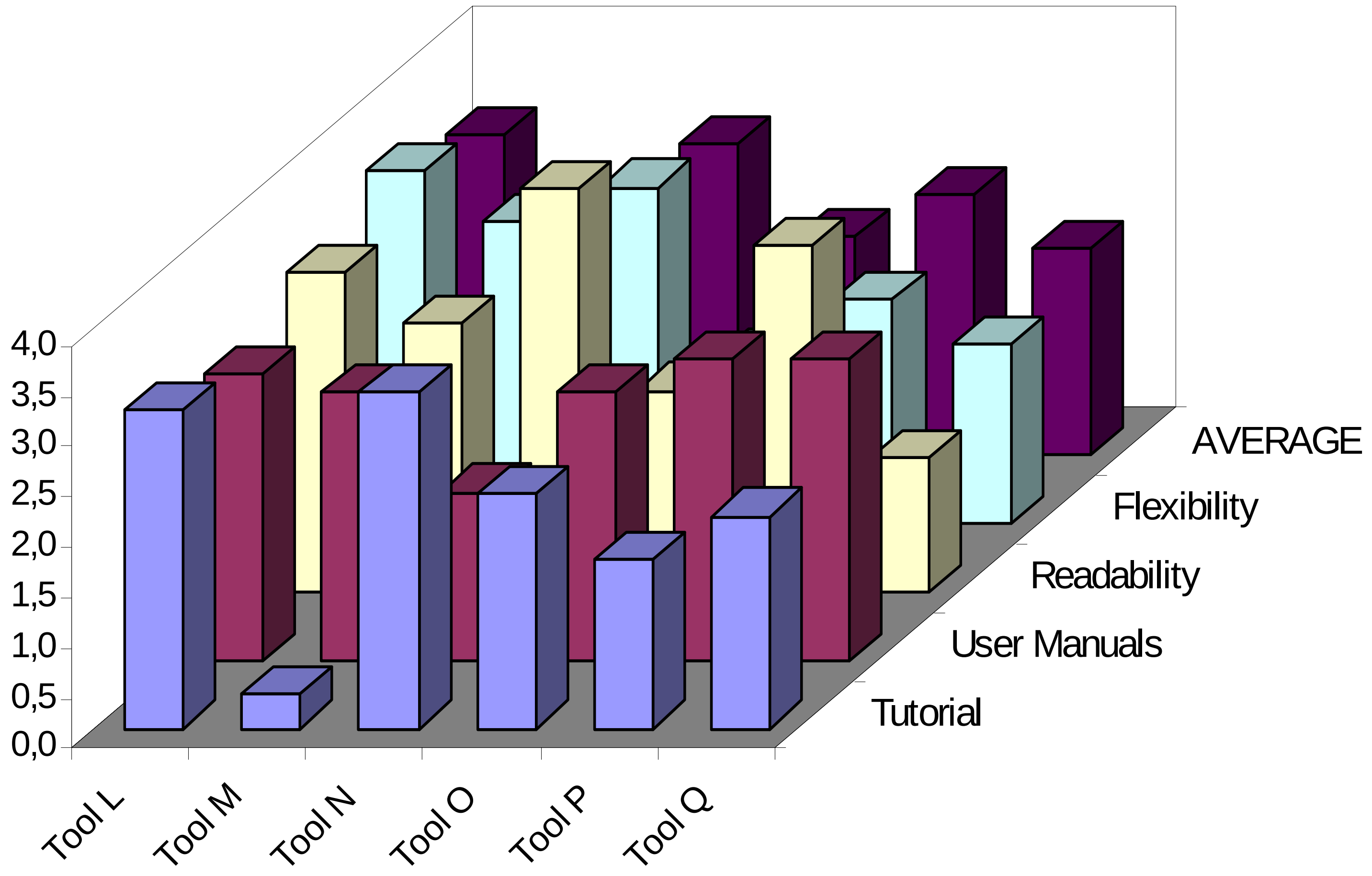
Controlled Experiment: Usability





Real Time Safety

Controlled Experiment: Functionality



Real Time Safety

Controlled Experiment: Traceability

- Small system size (only ten requirements) allowed developers to:
  - identify and manually trace all requirements to their equivalent modeling components
  - identify and manually trace all modeling components to their equivalent code components
  - identify and manually trace all code components to their equivalent modeling components
- For all tools:
  - the model and code components matched
  - the experiment identified code components generated as the run-time framework with no direct trace to the requirements

Experiments: Viewpoints

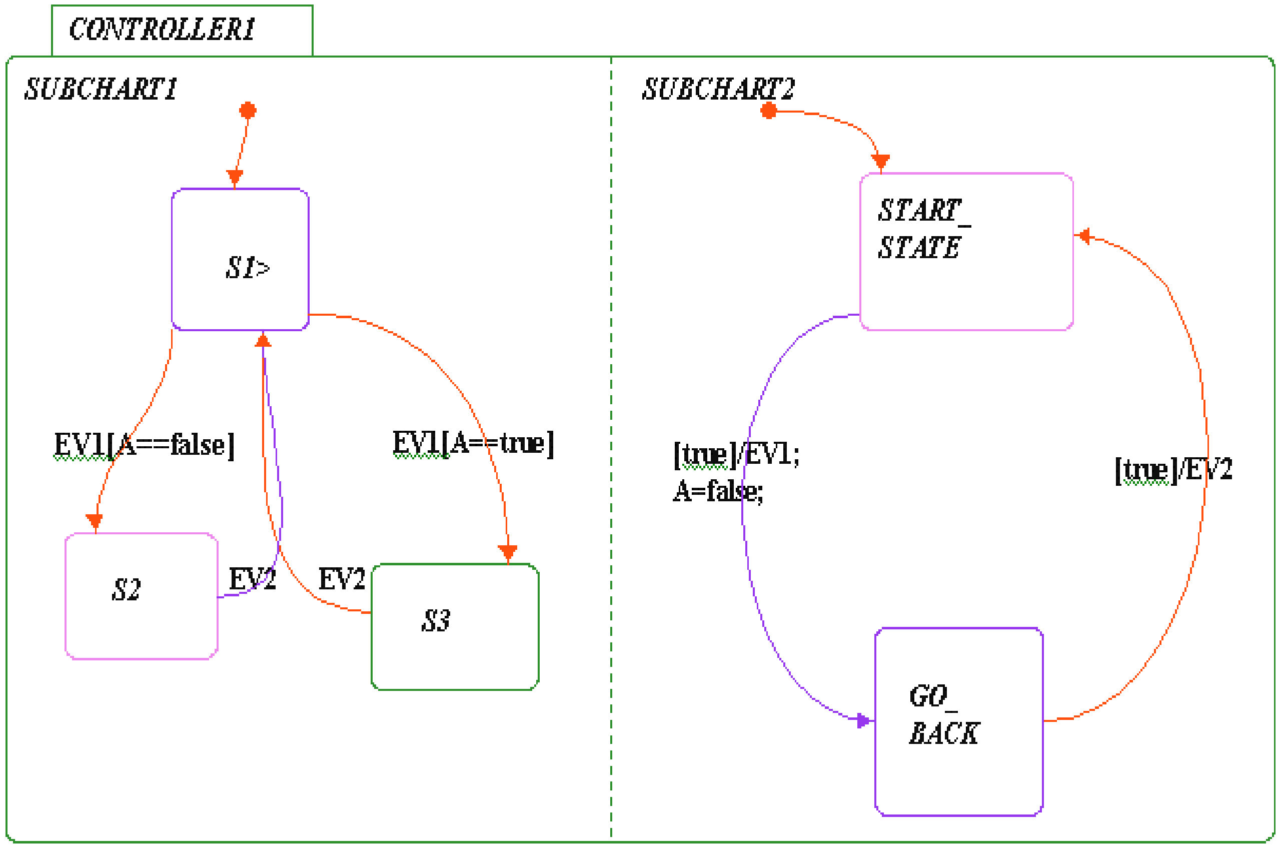
- Software Engineering Viewpoint (Discrete Models):
  - **Interactive** systems which use asynchronous languages based on interleaving tasks and operating systems principles (viewed as non-deterministic)
- Control Engineering Viewpoint (Continuous Models):
  - **Reactive** systems using event sequencing and logical time abstraction on common discrete time scale computing one step at the time (considered to be deterministic)
    - Formal - by reducing the system to set of dynamic equations
    - Practical - by modeling and solving differential equations

Experiments: Determinism

- Restrictive interpretation of determinism: the same input necessarily leads to exactly the same output
- More accurate interpretation of determinism for tools: established the ability to determine correctness of the output from the tool
- Approach:
  - Construct a state machine using variation of build sequences (the states and transitions built in different order)
  - Analyze the code generated from these variable builds
- The experiments show that the generated code may have different structure, but provides the same behavior

Behavioral Experiment: Model

- Create simple "producer-consumer" state machine
- Implement the model using various tools
- Test the behavior of generated code



Real Time Safety

Behavioral Experiment: Results

Tool	Behavior
TOOL X	Upon receiving and consuming EV1, the SUBCHART1 alternated between entering S2 and S3. The specific timing of the tool in resetting A to TRUE allowed the entry to state S2, even though it was not intention of the original design
TOOL Y	With the guarded transitions between states in the SUBCHART1, transitions occurred between states in the SUBCHART2 only, and the events are generated, but EV1 and EV2 were never consumed. No transitions from S1 in SUBCHART1. No entry into S2 nor S3 ever occurs
TOOL Z	With the guarded transitions in the SUBCHART1 the events EV1 and EV2 would *occasionally* be consumed. Several transitions would occur in SUBCHART2 before either EV1 or EV2 would be consumed. State S2 was never reached. Transitions were only between states S1 and S3

Real Time Safety

Experiments: Synchronicity

- The **synchronous** approach enables to achieve **determinism** and is suited for **time-triggered** applications/functions
- It is based on the abstract viewpoint that programs instantaneously and deterministically react to input events coming from their environment
- The programs cyclically (in the loop):
  - read inputs (events & values)
  - compute the systems outputs and/or new states
  - write the outputs
- Generally the loop is performed according to a basic clock; some computations may be performed at a lower pace (for instance every two cycles)
- The approach used in qualified tools (SCADE)

Real Time Safety

Experiment: Comments

- Tools with code generators allow developers to focus on a higher level of abstraction rather than mundane coding
- However …
  - Specific tool's approach may constrain development methodology
  - Modern development tools require a long learning curve
  - Inadequate documentation and tutorials
  - Possible tool malfunction due to workstation environment
  - Unclear messages in code generation, compiling and downloading

Real Time Safety

Outline

- Project Background
- Research Approach
- Development Tool Assessment
- Development Tool Qualification
- Experiments
- **Software Tool Forum**
- Research Findings
- Conclusions

Real Time Safety

Software Tool Forum: ERAU Daytona Beach, FL

- May 18/19, 2004: 150 participants representing 68 organizations (industry, government, academia):

  Ada Core Technologies, Airbus, Aircraft Braking Systems, ATL Software, Altair Avionics, Ametek Aerospace, Aonix, Artisan Software, Avidyne Corp, Avionics, Avionyx, Inc., Avista Inc., BAE Systems, Barco, Bechtel, Belcan, Boeing, CertTech LLC, Cessna Aircraft, CMC Electronics, Comm-Nav, Crane Aerospace & Electronics, CS Canada Inc., Directorate of Technical Airworthiness, DiSTI, Embedded Plus, ERAU, EMBRAER, Engenuity Technologies, ENSCO, Inc., Escher Technologies, Inc., FAA, FGCU, Garmin, GB Tech Inc,. GE Aircraft Engines, General Aviation Manufacturers, Goodrich Corp., Green Hills Software, Gulfstream Aerospace, Hamilton Sundstrand, High Integrity Solutions, Honeywell, IAI, Iowa State University, LDRA Technology Inc., Lockheed Martin, Metrowerks, MPC Products, NASA Langley, Polyspace, Pratt & Whitney, Real-Time Software Solutions, Rockwell – Collins, Rolls Royce, Safe Flight Instruments, Software Engineering Institute, Software Productivity Consortium, Teledyne Controls, MathWorks, MITRE Corp, Titan Systems, Transport Canada, TTTech, University of Minnesota, University of Paderborn, University of York, Vector Software, Verocel

- Identification of issues to consider in the future actions and need for clarification/guidelines

Real Time Safety

Software Tool Forum: Priority List (1)

- **Development tool qualification criteria need to be modified**
  - Criteria for tool qualification too stringent
  - Differences between the tool and the target software
  - Guidance for COTS missing
  - Platform issues
  - Partial credit would be needed
  - Flexible rigor for tool qualification
  - Code reviews issues
- **Criteria needs to be established to fit the Model-Based Development (MBD)**
  - Lifecycle consideration
  - Definition of what is the “source code”
  - Fuzzy boundary between requirements and implementation
  - Structural coverage for models
  - How to analyze the models?
  - Thorough analysis for the generated code?
  - Guidelines for model reviews

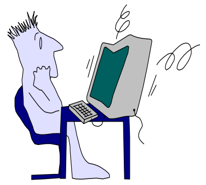
Software Tool Forum: Priority List (2)

Real Time Safety

- **Qualification criteria need to be developed that enable qualification to be carried from one program to another**
  - Definition of certification vs. qualification
  - Reuse credit for the tool software
  - Formal qualification approval document
  - Tool upgrades and their impact on qualification status
  - Specific list of documents required for qualification credit
- **Automatic Code Generator (ACG) usage and qualification require developing and documenting different approaches**
  - Specific qualification process for ACG technology
  - Is automatically generated code review practical?
  - Do we need to qualify ACG? (we do not do it for compilers)

Real Time Safety

Outline



Real Time Safety

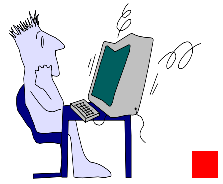
Research Findings: Industry View

- **Basic Issues**
  - Reliance on verification
  - Simplicity of in-house tools
  - Intellectual property issue for COTS
- **Qualification Process**
  - Current guidelines restrictions
  - “Business as usual” vs. trust in advances of software engineering technology
- **Acceptance of Qualification Approaches**
  - What is “source code” in MBD paradigm?
  - Development tool “determinism”?
  - Need for tool testing and verification
  - Roadways: the simplicity of tool function, separation of concerns, partitioning, use of model checking and formal evaluation

Real Time Safety

Research Findings: Industry View (cont)

- **Safe Usage of Tools**
  - Education and training
  - Tool constraints and limitation
  - Lifecycle with safety closely interfaced process
  - Strict tool version control
  - Tool platforms and operating environments
- **Kinds of Development Tools**
  - State of the art ahead of the regulations
  - Futuristic features vs. quality and applicability
  - Design properties assurance
  - Easy interface
- **Tools Considered for Qualification**
  - Simple automatic transformation utilities
  - Intellectual property and data ownership
  - COTS design tools with ACG capability



Research Findings: Qualification

- **Why do we need to Qualify Development Tools?**
  - Reduction of development and certification effort
  - Focus on higher level of abstraction
  - Reuse
- **What Tools Were Attempted to be Qualified**?
  - Few in-house tools qualified on certified projects *(CLARA, GALA, CTG, GPU, UTBT)*
  - COTS qualified by European JAA: *SCADE KCG (Esterel Technologies) and VAPS CG (eNGENUITY)*
  - Interest to qualify functional/block-oriented tools *(Mathworks, TNI-Software, Applied Dynamics, and National Instruments)*
  - Limited interest to qualify the structural/object-oriented tools *(iLogix, TNI-Valiosys, IBM/Rational, and Artisan Software)*

Real Time Safety

Research Findings: Qualification (cont)

- **How to Achieve Qualification for COTS Tools?**
  - DO-178B is revision: stand-alone tool qualification?
  - Version control, precise definition of operational environment, constraints, and limitations
  - Availability of tool software development data
- **Development Tools Qualification Barriers**
  - The state of regulations and guidelines
  - The business model, lack of incentives, cost
  - Lack of comprehensive tool data
  - Certification is the goal
- **Factors Regarding Tool Qualification**
  - Identification of metrics
  - An independent qualification lab?
  - Development tool information disclosure?
  - FAA-sponsored database?

Real Time Safety

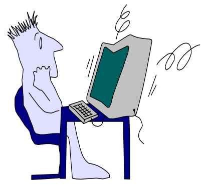
Research Findings: Quality Assessment

- **Assessment of Quality for Safety/Real-Time**
  - Development tools are software artifacts
  - Software development strategies for safety
  - Target application vs. development tool software
- **Evaluation Mechanisms and Methods**
  - Product: functionality and quality of service
  - Process: following defined procedures
- **Evaluation Criteria**
  - Tool selection
  - Standards compliance
  - Criteria adoption
- **Log-Term Support**
  - Tool vendors vs. regulated industry
  - Software upgrades, updates, and obsolescence
  - Software industry instability

Real Time Safety

Research Findings: Evaluation Taxonomy

- **Tool Functionalities** (design tool / ACG)
  - Graphical model representing system properties
  - Validation, simulation and animation
  - Automatic translation and target interface
  - Document generation and the version control
- **Tool Categorization**
  - Reflection of the software lifecycle phases
  - Focus on the composition and behavior
  - Model quality and transformation correctness
- **Categories/Function Vital for Development**
  - Model Driven Development paradigm
  - ACG interest: quality of the translation
- **Categories/Functions the Need to be Evaluated**
  - Code generation
  - Complexity of modern tools
  - Qualification of narrow functionality of the tool



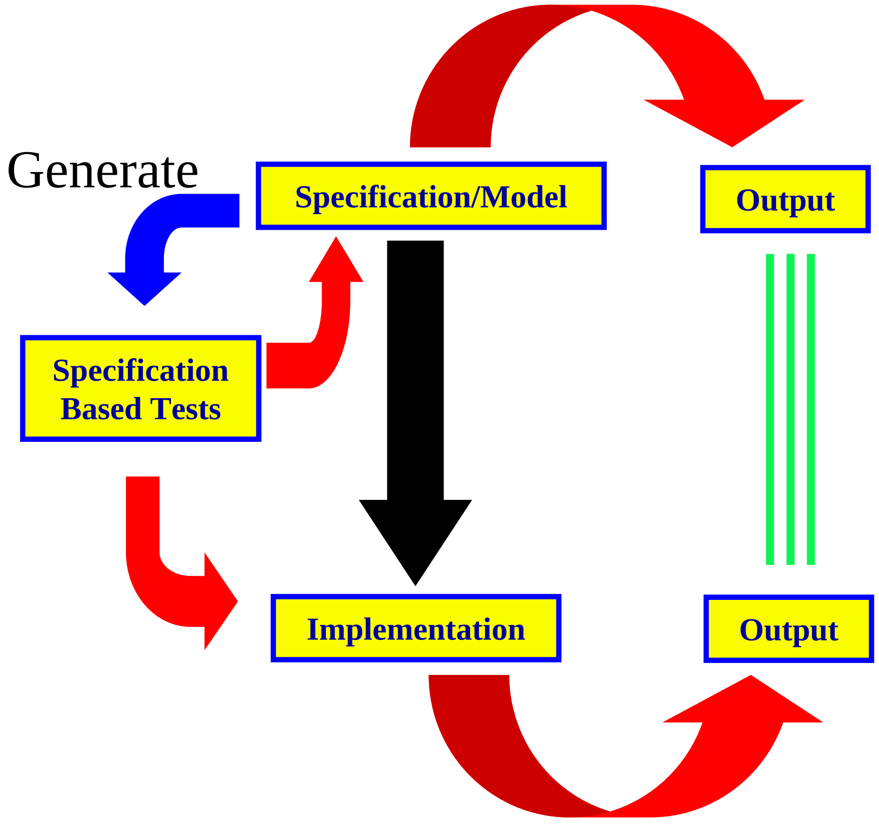
Real Time Safety

Outline

- Project Background
- Research Approach
- Development Tool Assessment
- Development Tool Qualification
- Experiments
- Software Tool Forum
- Research Findings
- **Conclusions**

Conclusions:
How to Show Correctness of Code Generation?

- Generate test suites from specification
  - Formal representation for specification model
  - Compare specification behavior with generated code
  - Automate, not eliminate, unit testing
- The match of two outputs makes an argument on correctness of generated code



{source: Whalen and Heimdahl}

Real Time Safety

Conclusions: COTS Tool Qualification Strategy

- Use existing software lifecycle items:
  - Tool source code
  - Tool design documentation (headers and comments)
  - Configuration management and SQA policies
- Reverse engineer (reconstruct):
  - Tool Software Requirements Specification
  - Tool Software Development Plan
  - Tool Software Verification Plan
  - Tool Software Design Document
  - Tool Traceability Matrix
  - Tool Software Accomplishment Summary
  - Tool Software Test Results

Real Time Safety

Conclusions: Challenges for Development Tools

- How to Validate Models?
  - The models must satisfy the requirements
  - The properties used in analysis must hold
  - Model testing is of paramount importance
- How to Validate Tools?
  - We will rely a lot on tools for model validation, can we trust them?
  - Creative use of testing necessary
- How to Verify and Validate Generated Code?
  - Can we trust that the translation was correct?
  - Test automation is critical
  - Includes output to analysis tools
- How to Handle Tool Evolution?
  - Tools will evolve and change—re-qualification
  - Models will not come in one language—translation

Conclusions: Final Remarks

- New technologies and tool capabilities
- Model Based Development paradigm
- Cost of tool qualification
- Partial certification credit
- How to handle tool evolution?
- Tool audition independent of application?
- Update DO-178B/ED-12B
- Develop a new guidance document dedicated to software tools
- Define research tasks to address specific tool issues

Real Time Safety

Publications

- Kornecki, A., Zalewski, J., (2005), “Experimental Evaluation of Software Development Tools for Safety Critical Real-Time Systems” accepted in **NASA Journal Innovations in Systems and Software Engineering**, June, 2005
- Kornecki, A., Zalewski, J., (2005), “Software Development Tool Qualification from the DO-178B Certification Perspective” accepted in **Crosstalk: the Journal of Defense Software Engineering**, July, 2005
- Kornecki, A., Zalewski, J., (2005), “Process-Based Experiment for Design Tool Assessment in Real-Time Safety-Critical Software Development”, to be printed in proceedings of 29th Annual **IEEE/NASA Software Engineering Workshop**, April 2005
- Kornecki, A., Hall, K., (2004), “Approaches to Assure Safety in Fly-by-wire Systems: Airbus vs. Boeing”, Proceeding of the Eight **IASTED** International **Conference on Software Engineering and Applications** (SEA 2004), ISBN: 0-88986-427-6, MIT, Cambridge, MA, November 2004, CD-edition
- Kornecki, A., Zalewski, J., (2004), “Criteria for Software Tools Evaluation in the Development of Safety-Critical Real-Time Systems”, Proceeding of 28th **PSAM7-ESREL’04** Conference, Berlin, Germany, June 2004, pp. 2364-2370
- Kornecki, A., Erwin, J. (2004), “Characteristics of Safety Critical Software”, Proceedings of the 22nd International **System Safety Conference**, System Safety Society, ISBN 0-9721385-4-4, Providence, RI, August 2004, CD-edition
- Kornecki, A., Hall, K., Hearn, D., Lau, H., Zalewski, J., (2004), “Evaluation of Software Development Tools for High Assurance Safety Critical Systems”, fast Abstract in Proceedings of 8th IEEE International **Symposium on High Assurance Systems Engineering**, HASE’04, ISSN 1530-2059, March 2004, pp. 273-274
- Crawford, L., Erwin, J., Grimaldi, S., Mitra, S., Kornecki, A., Gluch, D., (2004), “A Study of Automatic Code Generation for Safety-Critical Software: Preliminary Report”, fast Abstract in Proceedings of 8th IEEE International **Symposium on High Assurance Systems Engineering**, HASE’04, ISSN 1530-2059, March 2004, pp. 287-288
- Kornecki, A., Zalewski, J., (2003), “Design Tool Assessment for Safety-Critical Software Development”, Proceeding of 28th **IEEE/NASA Software Engineering Workshop**, December 2003, pp. 105-113
- Kornecki, A., Zalewski, J., (2003), “Assessment of Software Development Tools for Safety Critical Real Time Systems” Invited Paper in **IFAC Workshop on Programmable Devices and Systems**, Ostrava, Czech Republic, February 2003, pp. 2-7